TTAP-CSC-FER-00-002

SYBASE, INC.
ADAPTIVE SERVER ANYWHERE 7.0.0
WITH C2 UPDATE
FINAL EVALUATION REPORT

Kristina Rogers
Michael Boberski
Victoria Ashby

12 September 2000

Suite 100 West ♦ 7927 Jones Branch Drive ♦ McLean, VA 22102-3305 ♦ 703 848-0883 ♦ Fax 703 848-0960

TABLE OF CONTENTS

TABLE OF FIGURES AND TABLES

1 Introduction

1.1 Identification

This report documents the evaluation of Sybase, Inc.'s Adaptive Server Anywhere 7.0.0 Database Management System (DBMS) running on Microsoft's Windows NT 4.0 operating system. Adaptive Server Anywhere 7.0.0 with C2 Update (ASA) was evaluated against the C2 requirements of the Trusted Computer System Evaluation Criteria (TCSEC) as interpreted by the Trusted Database Interpretation.

1.2 Product Background

Adaptive Server Anywhere 7.0.0 enables the design and delivery of information to workgroup, mobile, and embedded database systems. ASA delivers rich database functionality, including full transaction processing, stored procedures, row-level locking, Java support, and symmetric multiprocessor (SMP) support, but still has a very small footprint.

The evaluated configuration is Sybase, Inc.'s Adaptive Server Anywhere 7.0.0 with C2 Update (ASA) running on Microsoft's Windows NT 4.0 with Service Pack 6a with C2 Update. The evaluated ASA software includes both the network and stand-alone versions of the server software (dbsrv7.exe and dbeng7.exe). However, the ASA server cannot be accessed remotely in the evaluated configuration. The client and server were evaluated running on the same machine. The trusted client interfaces include Interactive SQL, command line utilities, and the C programming language interface. Communication between client applications and ASA in the evaluated configuration occurs through NT named pipes, which are protected by NT access control mechanisms. Identification and authentication is performed using integrated login in the evaluated configuration. The ASA integrated login mechanism maps NT accounts to ASA user names.

The ASA evaluated configuration runs on the NT 4.0 evaluated configuration as specified in the *Microsoft Windows NT 4.0 Final Evaluation Report*. ASA does not depend on any special configuration of NT's kernel, executive, or other system software. The evaluated configuration does require that NTFS, NT's native file system, be used to format the hard disk that stores the ASA executables and user data.

1.3 Evaluation Status

Adaptive Server Anywhere began its evaluation under the Trust Technology Assessment Program (TTAP) in October of 1998 by the CygnaCom Solutions TTAP Evaluation Facility. The evaluation completed its Test Technical Review Board (TRB) in May of 1999. Then the evaluation was put on hold for several months awaiting the completion of the Microsoft Windows NT 4.0 evaluation. The evaluation restarted in February 2000. Security testing was completed at Sybase's Waterloo, ON facility on 7 April 2000. The Final TRB was held on 7 June 2000.

1.4 TTAP Evaluation Process Overview

This section describes the TTAP evaluation process. The following text is copied from the TTAP Process document.

"The TTAP evaluation process consists of three high-level phases:

- Pre-Evaluation,
- Evaluation,
- and Rating Maintenance.[1]

"Each of the phases and their associated activities are discussed in greater detail in subsequent chapters. The following subsections provide a brief overview of the TTAP process and highlight the activities associated with each phase.

1.4.1 Pre-Evaluation

"Pre-Evaluation consists of those activities that are recommended for a TTAP Evaluation Facility (TEF) to complete prior to beginning an evaluation to ensure that the product and its associated evaluation evidence are ready for evaluation. An inadequately tested product or incomplete documentation can substantially delay the schedule and increase the cost of an evaluation. Because the pre-evaluation activities occur only between the TEF and the vendor, TTAP imposes no requirements during this phase.

1.4.2 Evaluation

"Evaluation begins with a formal agreement between the vendor and the TEF. All contract negotiations, including any discussions regarding the price of an evaluation, the nature or conditions of payment, and the schedule for the evaluation, are left entirely up to the TEF and the vendor. After the vendor and the TEF sign a contract, the TEF submits the name of the vendor, the name and type of product, the schedule, and the members of the evaluation team to the TTAP Oversight Board. The evaluation may begin as soon as the contract is signed. Note that no authorization to proceed is needed from the Oversight Board; the notification is simply to allow the board to allocate and schedule TRB resources.

"Work on the evaluation itself begins after the agreement is signed. The evaluation team must determine that the system meets all of the TCSEC C2 requirements, and prepare and defend an Initial Product Assessment Report (IPAR) documenting this determination. Since such a determination requires that the team understand the system to a level at which such analysis can be made, evaluation evidence in the form of design documentation, training, informal vendor presentations, etc., is provided by the vendor to the evaluation team. This process typically begins when the vendor provides the evaluation team with product documentation (design, test, user, etc.) and system-level, developer-oriented training for the vendor's product. Training is followed by a

[1] Not applicable to this product.

comprehensive review of the system design by the evaluation team. The team performs security analysis of the product design, including both hardware and software components of the system. The team analyzes the system design and reviews the user, test, and Rating Maintenance[2] documentation. The information gathered during design analysis is used to write an Initial Product Assessment Report (IPAR), which the team presents to a Technical Review Board (TRB). The team also briefs the TRB on the vendor's and the team's plans for testing the product.

"After the IPAR/Test TRB meeting, the evaluation team performs security testing on the product. The test results are combined with the IPAR and edited to form the Final Evaluation Report (FER). The evaluation team presents final responses to open issues and the results of testing to a Final TRB. The TRB makes its recommendations to the TTAP Oversight Board regarding the product's requested trust rating. The Oversight Board then makes the final decision to place the evaluated product on the EPL at the requested level. Finally, the EPL entries and the final evaluation reports are published, and the evaluation documentation is archived."

1.5 Report Organization

This report follows the format prescribed in the TTAP Process Document and contains the following chapters and appendices:

- Chapter 1 is an introduction.
- Chapter 2 provides an overview of the composite DBMS/OS Trusted Computing Base from the perspective of the Trusted Database Interpretation.
- Chapter 3 describes the OS environment and ASA's OS dependencies.
- Chapter 4 describes the DBMS architecture. It identifies the TCB components and describes the TCB interfaces. It provides an overview of the product with a emphasis on the security relevant aspects of the system architecture.
- Chapter 5 identifies and describes the subjects and objects on which the product's security policy is enforced.
- Chapter 6 describes the implementation of the C2 functional requirements.
- Chapter 7 describes the C2 assurance mechanisms of system architecture and security testing.
- Chapter 8 addresses each of the C2 TCSEC/TDI requirements one at a time.
- Chapter 9 addresses TCSEC interpretations.
- Appendix A identifies the evaluated configuration.
- Appendix B contains the Evaluated Products List entry.
- Appendix C consists of tables documenting the SQL statement interfaces and their associated Discretionary Access Control permissions.
- Appendix D documents the system stored procedures.

[2] This text is taken from the TTAP C2 Process Document; however, Sybase has opted not to participate in the Rating Maintenance Program.

- Appendix E contains tables documenting the DBLIB functions.
- Appendix F lists acronyms.
- Appendix G contains the bibliography and references.

1.6 Conventions

SQL statements such as CREATE, GRANT, and SELECT are UPPER CASE.

Database utilities such as **dbisqlc**, **dblog**, and **dbtran** are **bold lower case**.

Permissions such as *SELECT, INSERT, UPDATE, DELETE, EXECUTE, ALTER,* and *REFERENCES* are *UPPER CASE ITALICS.*

System tables such as SYSTABLE and SYSLOGIN are UPPER CASE.

A user with DBA authority is referred to as the DBA. (DBA is also the ASA User ID that is automatically created when a database is created and that user has DBA authority.)

Procedures and Functions are both Named Objects in ASA. A Function is a Procedure that returns a value. Procedures and functions behave similarly with respect to the C2 security policy. All statements in this document regarding ownership, access controls, and audit that apply to procedures also apply to functions. Therefore, the term “procedures” is used in the text as a generic term for both procedures and Functions.

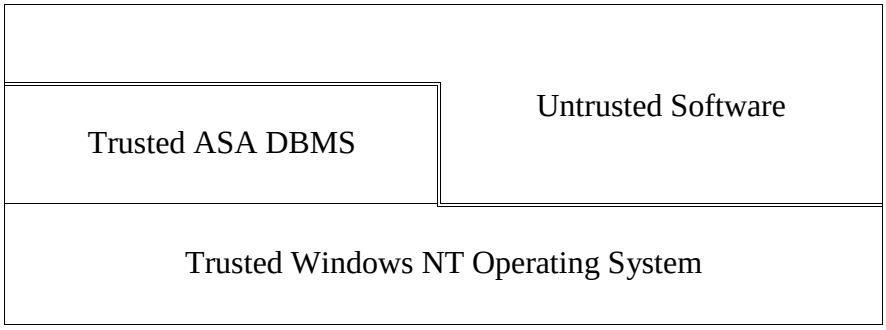
Windows NT, NT 4.0, NT, the operating system, and the OS all refer to Microsoft Windows NT® 4.0 with SP6a with C2 Update.

2 TCB Architecture

The Adaptive Server Anywhere (ASA) Trusted Computing Base (TCB) is a composite TCB made up of the ASA relational Database Management System (DBMS) and the Windows NT operating system (OS). The evaluated configuration consists of one or more instantiations of ASA running in a client-server environment on a single Windows NT platform.

The OS is the more primitive TCB subset because ASA relies on the OS to enforce portions of its security policy and to provide ASA with operating system services and storage objects to implement the DBMS and its objects. Figure 2.1, Composite TCB, shows the relationship between the two TCB subsets. The remainder of this chapter provides an overview of ASA database management functionality and describes the security policy and architecture of the composite TCB. This report focuses on the DBMS subset and its interface to the OS. For a description of the OS subset, see the Microsoft Windows NT 4.0 Final Evaluation Report.

Figure 2.1 – Composite TCB



2.1 ASA Database Management Functionality

ASA is a relational database management system. The relational data model represents data in a database as a collection of tables or relations that are accessed through a query language. ASA implements relations in a database as tables of data organized into columns and rows. Each column of a table stands for one attribute of the topic of that table. Each row of a table stands for one instance of the topic of the table and is made up of the set of values associated with the columns.

The relational model supports three fundamental operations: selection, projection, and join. The select operation retrieves a subset of rows in a table or tables. When issuing a select, a user specifies the selection criteria and the DBMS retrieves only the rows that meet those criteria. A projection on a table retrieves a subset of the columns from a table. Joins provide the mechanism for a requester to concatenate related information found in two or more tables.

ASA provides the ability to manipulate and manage databases and tables through the Structured Query Language (SQL). ASA supports the creation of indexes on tables for quicker access. ASA also supports creation and maintenance of views, which are logical tables derived from other tables. A view does not exist in physical form It is created through a stored SQL statement that performs selections, projections and joins on tables in a database to present a specific view of the database tables. Tables and views are named objects in ASA.

The other types of objects protected by ASA are procedures and functions. They are a set of SQL statements and may include program control statements such as conditional clauses and case statements. Parameters may also be passed in and out of a procedure or function. ASA supports both procedures and functions. A function is a specific type of procedure that returns a value. Procedures and functions can be used to enhance both performance and security. They can be compiled once and executed several times on the server. They can also limit user access to tables to a predefined set of operations.

ASA provides many features to support data integrity. Integrity constraints include check conditions, entity integrity, and referential integrity. Check conditions can be specified on columns to define the values or range of data that are acceptable for a given column. Entity integrity allows a unique, non-null primary key to be defined on a table. Referential integrity ensures that the value of a specified foreign key in one table matches the primary key in another or the same table. Another mechanism for ensuring integrity is triggers. Triggers can be defined to fire on an UPDATE, INSERT, or DELETE operation on a table. When a trigger is fired, it executes a set of SQL statements that can operate on the same or other tables. For example, a trigger on UPDATE on one table could cause a corresponding DELETE on another table.

ASA also supports transaction processing, locks and isolation levels. Transactions can be used to ensure that logically related SQL statements are executed as an atomic unit. Transactions support data recovery in the event of system failure and the interweaving of commands from concurrent users. ASA uses row level locking to allow transactions to execute concurrently with minimal interference. ASA implements read locks, write locks, and phantom locks. ASA supports the four standard isolations levels: levels 0 through 3. Level 0 implements the least locking and Level 3 implements the most locking. Isolation levels provide for a tradeoff between transaction consistency and concurrency. For more information on locking and isolation levels, see Section 4.15, Transaction Processing.

ASA provides a backup utility and three logs to support data recovery in the event of system or media failure: checkpoint log, transaction log, and rollback log. The checkpoint log is located in the database file and saves copies of updated pages before they are written out to disk. A checkpoint occurs when all the updated pages are written out to disk. The checkpoint log is deleted following a checkpoint. The transaction log records all changes to the database in the order that they occur. Inserts, updates, deletes, commits, rollbacks, and database schema changes are all logged. The TFM requires that an ASA database be run with a transaction log. If the database is run without a transaction log, a checkpoint is carried out whenever a transaction is committed. To provide

additional protection in the event of media failure, it is recommended that the database files and the transaction log be stored on separate devices. There is a separate rollback log for each connection and it records all changes made to the contents of tables. Rollback logs are deleted when a transaction is committed.

ASA provides interfaces both for interactive users and application programs. The interface for interactive users is Interactive SQL (ISQL). Embedded SQL (ESQL) provides a programming language interface. Application programs can use cursors to control the fetching of rows from a table. A cursor is a symbolic name that is associated with a SELECT statement or stored procedure that returns a result set. There are ESQL commands to declare, open, fetch, and close cursors.

2.2 Composite Trusted Computing Base (TCB) Security Policy

The composite TCB enforces a single security policy that does not allow users to access information for which they have not been granted authorization.

- **Subjects:** Subjects in the OS are Windows NT processes; subjects in ASA are connections. The ASA server runs as an NT 4.0 process owned by user Sybase.

- **Objects:** Information in the composite TCB is stored in objects. Access to information contained in these objects is mediated by the composite TCB. The subset of composite TCB objects provided by ASA are maintained within OS objects that are protected from non-ASA access using OS protection mechanisms.

- **Identification and authentication (I&A)**: ASA implements an integrated login capability that maps NT user accounts to ASA user names and that is required in the evaluated configuration.

- **Discretionary Access Control (DAC):** ASA is responsible for managing access to all DBMS objects. ASA DAC policy augments the OS DAC policy and applies to the DBMS objects: tables, views, procedures, and functions. Upon creation, only the owner of a DBMS object has access to the object. The owner of the object may grant other users and groups access to the object.

- **Audit:** ASA provides its own separate audit capability. Audit records are stored in files in a directory protected by NT 4.0 DAC.

- **Trusted Administrators**: NT 4.0 trusted administrators and ASA trusted users with DBA authority are created independently. ASA relies upon an NT 4.0 administrator to create users, to create directories, to set permissions, to install the ASA software, and to create, start, and stop the ASA service. The trusted user with respect to ASA is the Sybase user. The Sybase user is granted the "Logon as a Service" user right, but otherwise runs without any privileges. There are two ASA authorities in the evaluated configuration: DBA and Resource. DBA authority is for database administrators and must only be granted to trusted users. Resource authority gives a

user the ability to create objects in a database, but is not trusted with respect to the security policy. Two other authorities: Remote DBA authority and Publisher authority support replication, which is not included in the evaluated configuration. The TFM directs the administrator not to grant the Remote DBA authority in the evaluated configuration, and the Publisher authority does nothing without Remote DBA authority.

For more information about security policy relationships between ASA and the OS, see Chapter 3, Operating System Environment. For more details about ASA security mechanisms, see Chapter 6, TCB Exported Security Policies.

2.3 Composite TCB Architecture

The TCB encompasses both the OS TCB and the DBMS TCB. Cooperation between the two subsets ensures overall enforcement of the security policy.

- ASA clients and server run as separate processes on Windows NT 4.0. The ASA server process runs as an NT 4.0 Service under the "Sybase" NT account. Client processes are associated with the NT user account with which the user logs into NT.

- The Sybase user has the "Logon as a Service" user right, but otherwise does not run with any NT 4.0 privileges. The "Logon as a Service" user right is required to start up the ASA server as an NT service, but is not stored as a privilege in the Security Access Token for the ASA Server process.

- The DBMS uses the resources provided by the OS for proper functioning, but does not tamper with OS objects.

- The OS provides and protects several types of resources for the DBMS. The OS files containing the ASA executable are stored in the NT 4.0 C:\program files\sybase directory and are protected by the OS DAC mechanisms. ASA database objects and configuration data are stored in the C:\databases and C:\astmp directories, which are also protected by the OS DAC mechanism. The OS also provides memory to ASA and protects the memory from other OS processes. ASA uses the memory for internal processing and data structure storage.

- ASA protects its own internal data in its own system tables that are owned by the ASA user SYS. SYS is a trusted ASA user account to which no one can connect.

For more information about system architecture dependencies of ASA on NT 4.0, see Chapter 3, Operating System Environment and Section 7.1, System Architecture.

3 Operating System Environment

Adaptive Server Anywhere 7.0.0 with C2 Update (ASA) runs on the evaluated configuration of Microsoft Windows NT 4.0 Service Pack 6a with C2 update. ASA was evaluated with the client and the engine or server running on the same machine communicating over named pipes.

This section addresses the operating system dependencies of ASA on Windows NT 4.0. The following topics are addressed:

- Hardware,
- NT Evaluated Configuration,
- Users and Groups,
- Directories and File Permissions,
- User Rights,
- Running as a service,
- Audit,
- Integrated Login,
- Registry Keys,
- VGA Driver,
- Networking,
- Named Pipes,
- Administrative Tools,
- Windows NT 4.0 Application Programming Interfaces, and
- NT 4.0 Dynamic Link Libraries.

3.1 Hardware Platform

ASA was tested on a Compaq Proliant 7000 Server, one of the machines on which the Windows NT 4.0 operating system was evaluated.

3.2 NT Evaluated Configuration

ASA was tested running on the Windows NT 4.0 Server as an NT domain member. NT 4.0 Service Pack 6a with C2 Update was installed as directed in the NT 4.0 TFM.

3.3 Users and Groups

3.3.1 Sybase

The ASA service runs with the NT User ID of Sybase. The Sybase user is a member of the Users group only. The only user right given to the Sybase User is "Log on as a service". The Sybase user is defined locally. The NT 4.0 Sybase user is mapped to the ASA DBA username for integrated login.

3.3.2 NT 4.0 Administrator

The support of a local NT 4.0 administrator is required to create the Sybase user, to make the Sybase user a member of the NT 4.0 Users group, to grant the user rights to the Sybase user, to create and set permissions on directories, to install the ASA software, and to create, start, and stop the ASA service. An ASA trusted user is a user with DBA authority in the ASA database. ASA DBA authority is granted independently of the NT administrator group.

3.4 Directory and File Permissions

ASA must be installed on an NTFS drive. (C: is used for the drive in the pathnames below, but it could be another NTFS drive such as D: or E:.) Also, the executables and data files could be stored in other directories, as long as they could be appropriately protected. To reduce the chances for confusion or error, specific directory names are used here and in the Installation section of the C2 Supplement.)

The Sybase executables are stored under C:\Program Files\Sybase. This directory is owned by the NT administrator. The permissions on this directory and its sub-directories are set as follows:

- Sybase: Full Control
- Administrators: Full Control
- Everyone: Read

No permissions are set for any other users or groups.

The C:\ Databases folder is used to store database and transaction log files. This directory is owned by the NT administrator. The permissions on this directory and its sub-directory are set as follows:

- Sybase: Full Control
- Administrators: Full Control
- Creator Owner: Full Control
- System: Full Control

No other NT users or groups are granted access.

The C:\ ASTMP folder is used by the engine for temporary storage. This directory is also owned by the NT administrator. The permissions are the same as for the C:\Databases folder. The System ASTMP environment variable is set to C:\ASTMP by the NT Administrator during ASA installation. The ASA engine uses the ASTMP folder to store intermediate results during execution. If the ASTMP environment variable were not set, ASA temporary files would be stored in the C:\temp directory, which is not protected.

3.5 User Rights

The only user right granted to the Sybase NT account is "Log on as a Service." The only documented capability that this user right provides is that it allows a process to register with the system as a service. The TFM instructs the administrator to start the server as an NT service. Starting the server as a service ensures that an untrusted user cannot start the server and change the switches that set configuration options. The NT 4.0 TFM states that "Log on as a Service" may be granted to trusted users as needed. Sybase is a trusted user with respect to the ASA TCB subset. "Log on as a service" is not one of the user rights that is implemented as an NT 4.0 privilege and stored in the Security Access Token for the process. This user right only needs to be checked to start up the service. Then ASA runs as an unprivileged process.

3.6 Running as a Service

ASA must run as a service in the evaluated configuration. The executable files are:

- C:\Program Files\Sybase\SQL Anywhere 7\win32\dbeng7.exe for the Stand-alone Engine
- C:\Program Files\Sybase\SQL Anywhere 7\win32\dbsrv7.exe for the Network Server.

The Installation section of the *ASA C2 Supplement* instructs the administrator to create the ASA service to run as user Sybase. The ASA service needs access to the C:\databases directory to start up a database in the evaluated configuration; only the Sybase user, the NT administrators, and the System have access to this directory.

3.7 Audit

Sybase ASA does not depend on the NT 4.0 application log to store any of its audit events. However, the TDI requires that it be possible to correlate events in the NT 4.0 audit log with events in the ASA audit logs.

3.7.1 Transaction Log File

Audit events generated when the engine is running are stored in the transaction log file which is stored in the protected C:\Databases directory.

3.7.2 Auditing of Database Utilities

The audit events generated by database utilities that can run when the engine is not running and that cannot, therefore, be written to the transaction log by the engine are stored in another file named <dbname>.alg located in the same protected directory, C:\Databases, where database files and the transaction log are stored.

3.7.3 Audit Correlation

To correlate NT audit events with ASA audit events, the ASA evaluated configuration requires integrated login.

3.7.4 Use of NT Audit

The NT audit mechanism can be used to audit logins to the NT Sybase account. During normal operations, this occurs when the ASA service is started. In addition, NT can be configured to audit access to the ASA data files. However, auditing file access in NT generates a large amount of NT audit data . The ASA data files are protected by NT DAC, so only processes running on behalf of the Sybase user (i.e., the ASA service) and NT trusted processes will be able to access these files. NT can also be configured to audit failed access attempts that could indicate an attempt to bypass the ASA security mechanisms.

3.8 Integrated Login

The ASA integrated login feature maps NT user accounts to ASA user IDs. The Installation section of the *ASA C2 Supplement* instructs the administrator to create a one-to-one mapping of NT user accounts and ASA user IDs. ASA uses the Security Support Provider Interface (SSPI) to verify the NT User ID. The NT system calls that ASA uses are as follows:

- AcquireCredentialsHandle,
- InitializeSecurityContext,
- CompleteAuthToken, and
- AcceptSecurityContext.

These system calls are in the NT evaluated configuration. The ASA Algorithm is as follows:

- The untrusted ASA client calls AcquireCredentialsHandle with a results buffer to obtain a handle to the reference credentials. Then the client calls InitializeSecurityContext, passing in the results buffer to obtain a security token that will be sent to the ASA Server. The return code from this function tells the client whether the authentication token needs to be completed. If so, the ASA calls CompleteAuthToken. This is required by some underlying transport protocols. The client sends the security token to the server. The return code from InitializeSecurityContext tells the client whether more messages are required. If so, the client waits for results from the server, putting them into the results buffer, re-initializing the results buffer and looping back to until done.

- The trusted ASA Server accepts the streams connection and receives the DU_CONNECT message. The server calls AcquireCredentialsHandle to obtain a handle to the reference credentials. The server calls AcceptSecurityContext, passing in the information provided by the client. The server verifies the client's user ID based on the results of the AcceptSecurityContext system call.

3.9 Registry Keys

During installation, ASA adds four of its own subkeys to the NT Registry:

- HKEY_LOCAL_MACHINE\SOFTWARE\Sybase\Adaptive Server Anywhere
- HKEY_LOCAL_MACHINE\SOFTWARE\Sybase\Sybase Central
- HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\Adaptive Server Anywhere 7.0
- HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\Adaptive Server Anywhere 7.0 translator

The Installation section of the *ASA C2 Supplement* requires that these keys be set to read only for everyone and full control for Sybase as part of the ASA installation.

The InstallShield program is automatically run during ASA installation. InstallShield runs with the privileges of the NT Administrator, which allows it to update registry entries. It was found during testing that InstallShield recreates some NT registry subkeys that were deleted while installing NT 4.0 in its evaluated configuration. Therefore, part of the ASA installation process is to manually re-delete the following subkeys after InstallShield has finished executing:

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\OS/2 Subsystem for NT

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Environment\Os2LibPath

The ASA installation process does not change any permissions on existing registry keys.

3.10 VGA Driver

The only graphics controller in the Microsoft Windows NT 4.0 evaluated configuration is vga.sys, which only provides for 16 colors. Providing only a 16 color graphics controller limited the ASA functionality that could be tested on the Windows NT 4.0 evaluated configuration and included in the ASA evaluated configuration. (Specifically, Sybase Central was not included in the ASA evaluated configuration, because it requires a 256 color VGA driver.)

3.11 Networking

Guidance is provided to the ASA Administrator in the Installation section of the *ASA C2 Supplement* to disable ASA networking functionality. This is done by creating databases using the -i switch of the **dbinit** command that disables jconnect support (which uses TCP/IP) and starting up the database server process with the -x namedpipes switch that limits communication in the evaluated configuration to named pipes.

It is acceptable in the evaluated configuration to have NT networking enabled, and there are no constraints concerning other machines on the network. Other machines will not be able to access the ASA server directly over the network. However, a user or a program on another machine could potentially start up an application on the ASA machine that connects to ASA as a client. The connection to the application program from the remote machine would be outside of the ASA evaluated configuration.

3.12 Named Pipes

As described in Section 3.11, Networking, ASA clients communicate with the ASA server over NT Named Pipes in the evaluated configuration.

3.13 Administrative Tools

The NT administrative tools that are required to install ASA in its evaluated configuration are listed in Table 3.1. Since InstallShield was not part of the NT 4.0 evaluated configuration, team tests were conducted on InstallShield to verify its operation and to make any necessary updates to the Installation section of the *C2 Supplement* about its use.

Table 3.1 - NT Administrative Tools

NT Operation	Administrative Tool	In NT TCB
Create user	User Manager	Yes
Assign password	User Manager	Yes
Set password options	User Manager	Yes
Assign user rights	User Manager	Yes
Set audit events	User Manager	Yes
Create folder	Windows NT Explorer	Yes
Change folder	Windows NT Explorer	Yes
Copy file	Windows NT Explorer	Yes
Install Sybase	Install Shield	No - Team Test
Set permissions on registry entries	Registry Editor	Yes
Set permissions on directories and files	Windows NT Explorer	Yes
Start/Stop Service	Control Panel/Services	Yes

3.14 Windows NT Application Programming Interfaces

ASA uses system calls and interfaces provided by NT, which are listed and described in Tables 3.2 through 3.6. These system calls provide various functions to ASA, including file I/O, network I/O, process management, get environment information, and other miscellaneous functions, including security functions, character set functions, memory management, string manipulation, error handling, and NT services management.

This section contains a list of all of the security-relevant Windows NT APIs used by ASA. They are broken down into different sections by broad functionality areas.

Table 3.2 - Windows NT File I/O System Calls

System Call	Purpose
CreateFile	Create a file object representing a file or directory.
CreateFileMapping	Map a file into memory.
CreateIOCompletionPort	Create an I/O completion port associated with a file.
DosDateTimeToFileTime	Converts DOS date/time to a 64-bit file time.
FileTimeToDosDateTime	Converts a 64-bit file time to DOS date/time format.
FileTimeToLocalFileTime	Converts a UTC-based file time to a local file time.
FileTimeToSystemTime	Converts a 64-bit file time to system time format.
FindClose	Closes a search handle.
FindFirstFile	Searches a directory for a file.
FlushFileBuffers	Flushes buffers associated with a file to the disk.
GetDiskFreeSpace	Returns the amount of free space on a disk.
GetDriveType	Returns the type (CD, hard disk, removable, etc.) of a disk.
GetFileInformationByHandle	Returns information on an open file.
GetFileSize	Returns file size.
GetFileTime	Returns file date/time.
GetFullPathName	Returns the full pathname of a file.
GetOverlappedResult	Returns the result of an asynchronous file I/O.
GetQueuedCompletionStatus	Removes an I/O packet from a completion port.
InitializeSecurityDescriptor	Initializes a security descriptor for a file.
IsBadReadPtr	Verifies read access to a memory address.
IsBadStringPtr	Verifies read access to a memory address containing a string.
LocalFileTimeToFileTime	Converts a local file time to a UTC-based file time.
MapViewOfFile	Maps a file into the address space.
OpenFileMapping	Opens a named file mapping.
PostQueuedCompletionStatus	Posts an I/O packet to a completion port.
ReadFile	Reads from a file.
ReadFileEx	Reads from a file asynchronously.
SetEndOfFile	Sets the end-of-file position for a file.
SetFileAttributes	Sets the attributes of a file.
SetFilePointer	Sets the file position of an open file.
SetFileTime	Sets the file time.
UnmapViewOfFile	Unmaps a mapped view of a file.
WriteFile	Writes to a file.
WriteFileEx	Writes to a file asynchronously.

1

2 **Table 3.3 - Windows NT Process Management System Calls**

System Call	Purpose
ConvertThreadToFiber	Converts a thread into a fiber.
CreateEvent	Creates an event.
CreateFiber	Creates a fiber object.
CreateMutex	Creates a mutex object, used for mutual exclusion.
CreateSemaphore	Creates a semaphore object.
DeleteCriticalSection	Deletes a critical section object.
DeregisterEventSource	Closes an event source handle.
EnterCriticalSection	Waits for ownership of a critical section.
GetCurrentProcess	Returns a process handle.
GetCurrentProcessId	Returns a process ID.
GetCurrentThread	Returns a thread handle.
GetCurrentThreadId	Returns a thread ID.
GetProcAddress	Returns the address of an exported function.
GetProcessAffinityMask	Gets the processor affinity mask for a process.
GetProcessTimes	Obtains timing information about a process.
InitializeCriticalSection	Initializes a critical section.
InterlockedExchange	Atomically exchanges a pair of 32-bit values.
KillTimer	Destroys a timer object.
LeaveCriticalSection	Gives up ownership of a critical section.
OpenProcess	Returns the handle of an existing process.
RegisterEventSource	Returns a handle that can be used to log an event.
RegisterServiceCtrlHandler	Registers a function to handle service control requests.
ReleaseMutex	Releases ownership of a mutex.
ReleaseSemaphore	Increases the count of a semaphore.
ReportEvent	Adds an entry to an event log.
ResumeThread	Resumes execution of a suspended thread.
SetEvent	Signals an event.
SetKernelObjectSecurity	Sets the security of a process, thread, or event.
SetPriorityClass	Sets the priority class for a process.
SetProcessAffinityMask	Sets the processor affinity mask for a process.
SetThreadPriority	Sets the priority class for a thread.
SetTimer	Creates a timer.
SuspendThread	Suspends execution of a thread.
SwitchToFiber	Schedules a fiber.
TlsAlloc	Allocates a thread local storage index.
TlsGetValue	Gets the value of a thread local storage index.
TlsSetValue	Sets the value of a thread local storage index.
WaitForMultipleObjects	Waits for any or all of a number of events to be signaled.
WaitForSingleObject	Waits for an event to be signaled.
WaitForSingleObjectEx	Waits for an event to be signaled, while allowing completion routines to run.

3

4 **Table 3.4 - Windows NT Named Pipes System Calls**

System Call	Purpose

ConnectNamedPipe	Waits for a client process to connect to a named pipe.
CreateNamedPipe	Creates a named pipe.
DisconnectNamedPipe	Disconnects a named pipe.
SetNamedPipeHandleState	Sets the read and blocking mode of a named pipe.
WaitNamedPipe	Attempt to connect to a named pipe.

1

2 **Table 3.5 - Windows NT Environment System Calls**

System Call	Purpose
GetEnvironmentVariable	Gets the value of an environment variable.
GetExitCodeProcess	Gets the exit code of a process.
GetLocalTime	Gets the current date and time.
GetModuleFileName	Gets the full path and filename for the executable file containing a module.
GetModuleHandle	Gets a handle for a module.
GetPrivateProfileString	Gets a string out of an initialization file.
GetSystemDefaultLangID	Gets the default language ID.
GetSystemDirectory	Gets the path of the Windows system directory.
GetSystemInfo	Gets information about the system.
GetSystemMetrics	Gets system configuration settings.
GetTickCount	Gets the number of milliseconds since Windows was started.
GetTimeZoneInformation	Gets the current time zone information.
GetUserName	Gets the user name of the current thread.
GetVersion	Gets the version of Windows.
GetVersionEx	Gets extended Windows version information.
GetWindowsDirectory	Gets the path of the Windows directory.
GlobalMemoryStatus	Gets information about system memory.

3

4 **Table 3.6 - Windows NT Miscellaneous System Calls**

System Call	Purpose
AcceptSecurityContext	Creates a security context using the opaque message (security token) that can be passed to the server.
AcquireCredentialsHandle	Acquires a handle to the reference credentials.
CharLower	Converts a string to lowercase.
CharNext	Returns a pointer to the next character in a string.
CharUpper	Converts a string to uppercase.
CloseHandle	Closes a handle.
CompleteAuthToken	Completes an authentication token, since some protocols, like DCE RPC, need to revise the security information once the transport has updated some message fields.
DuplicateHandle	Creates a copy of a handle.
FreeLibrary	Unloads a DLL.
GetACP	Returns the current ANSI code page.
GetLastError	Returns the error code for the most recent Windows API call.
InitializeSecurityContext	Initiates a security context by generating an opaque message (security token) that can be passed to the server.
IsDBCSLeadByte	Returns whether a character is a valid lead byte of a character in a double-byte character set.

System Call	Purpose
LoadLibrary	Loads a DLL.
Lstrlen	Returns the length of a string.
RegCloseKey	Releases the handle of a registry key.
RegCreateKeyEx	Creates or opens a registry key.
RegOpenKeyEx	Opens a registry key.
RegQueryValueEx	Returns the type and data for a value of a registry key.
RegSetValueEx	Stores data in the value field of a registry key.
SetErrorMode	Controls how the OS will handle certain types of errors.
SetLastError	Sets the last error code.
SetSecurityDescriptorDacl	Sets information in a discretionary access list.
SetServiceStatus	Updates the status of a service.
Sleep	Suspends execution for an interval.
SleepEx	Suspends execution for an interval, while allowing completion routines to run.
StartServiceCtrlDispatcher	Connects a service process to the service control manager.
VirtualAlloc	Reserves pages in the virtual address space.
VirtualFree	Releases pages in the virtual address space.

3.15 NT 4.0 Dynamic Link Libraries

This section documents the NT 4.0 dynamic link libraries used by ASA. Tables 3.7, 3.8, and 3.9 list the dynamic link libraries used by the engine, Interactive SQL, and DB tools respectively. All of these dynamic link libraries are in the NT evaluated configuration.

Table 3.7 - Windows NT Dynamic Link Libraries Used by the Server

USER32.dll
KERNEL32.dll
GDI32.dll
ADVAPI32.dll
RPCRT4.dll
SHELL32.dll
COMCTL32.dll
ole32.dll
NETAPI32.dll
MSVCRT.dll
NETRAP.dll
SAMLIB.dll
comdlg32.dll
WSOCK32.dll
WS2_32.dll
WS2HELP.dll
msidle.dll
MSVCRT40.dll
MSVCIRT.dll

Table 3.8 - Windows NT Dynamic Link Libraries Used by Interactive SQL

ntdll.dll
ADVAPI32.dll
KERNEL32.dll
USER32.dll
GDI32.dll
RPCRT4.dll
WSOCK32.dll
WS2_32.dll
MSVCRT.dll
WS2HELP.dll
NETAPI32.dll
NETRAP.dll
SAMLIB.dll
comdlg32.dll
SHELL32.dll
COMCTL32.dll
CTL3D32.dll
odbccp32.dll
ole32.dll
VERSION.dll
LZ32.dll
odbcint.dll
msidle.dll

Table 3.9 – Windows NT Dynamic Link Libraries Used by DB Tools

ntdll.dll
USER32.dll
KERNEL32.dll
GDI32.dll
ADVAPI32.dll
RPCRT4.dll
WSOCK32.dll
WS2_32.dll
MSVCRT.dll
WS2HELP.dll
NETAPI32.dll
NETRAP.dll
SAMLIB.dll

4 TCB Software

4.1 Architecture Overview

Adaptive Server Anywhere supports a client/server architecture. There are two versions of ASA:

- Stand-alone database engine, and
- Network database server.

Both share the same code; however, the stand-alone database engine does not support networking. Although the stand-alone database engine is designed to reside on a single user's machine, it can support up to ten simultaneous connections from one or more applications on the same machine.

The network database server can receive requests from client applications on other machines across a network. The network database server supports multiple concurrent connections and unlimited client communication links. The actual number of active concurrent connections permitted is determined by the license and the capacity of the network and server machine. The network database server is part of the evaluated configuration. The server was only tested with the client and the server running on the same platform, and the server cannot be accessed remotely in the evaluated configuration. However, the NT machine can be networked while ASA is running in its evaluated configuration.

For both versions, an ASA server executes as a single process that is wholly inside the TCB.

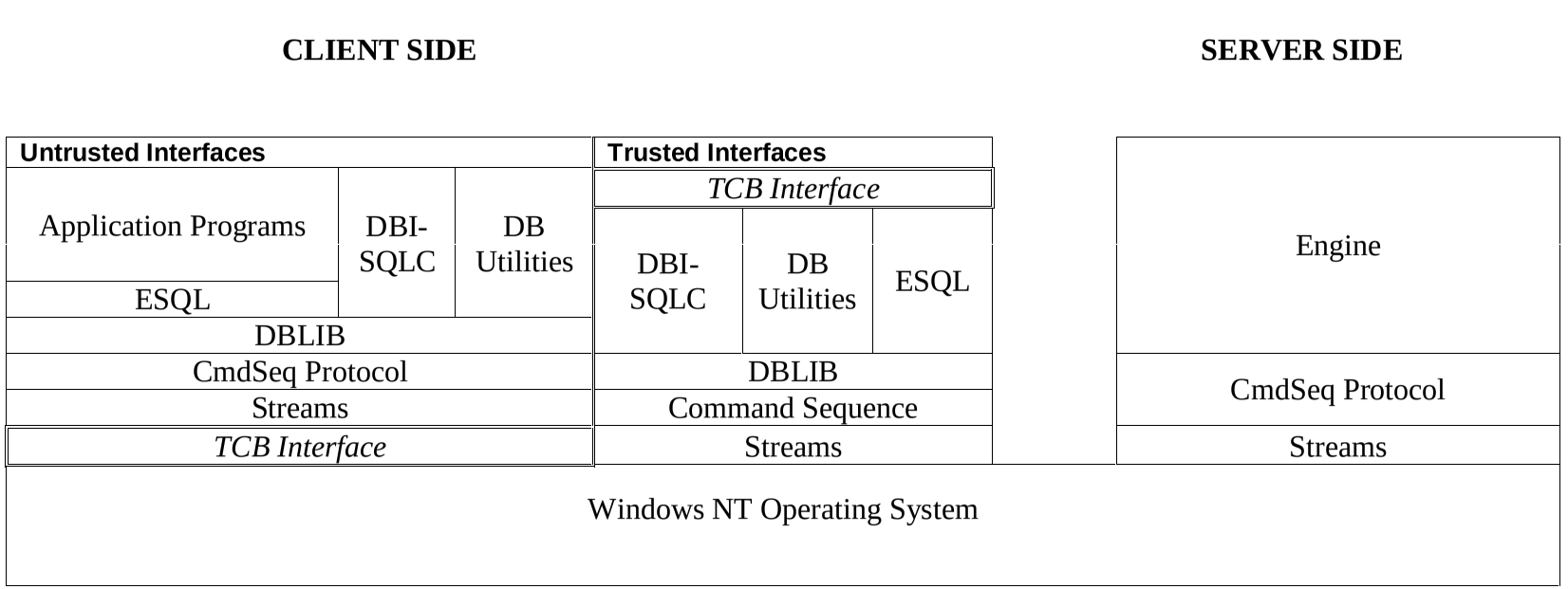
4.2 Client/Server Architecture

The client/server interface of the ASA consists of several layers. The lowest level is Windows NT named pipes. The next lowest level is the streams protocol. The highest levels are the application level interfaces of SQL and the administrative utilities. Figure 4.1 depicts the layering of the client/server interface and indicates where the TCB interface is for untrusted and trusted subjects.

4.2.1 Communications Layers

The communications layers: streams, the command sequence (CmdSeq) protocol and DB Library (DBLIB) are inside the TCB. These layers do not implement security functionality, but they must operate correctly. The streams, CmdSeq, and DBLIB layers are part of the trusted client. The streams, and CmdSeq layers are also server layers. These communications layers are described in Section 4.8, Communications.

Figure 4.1 - ASA TCB Interfaces



4.2.2 Untrusted Subject Clients

The far left of Figure 4.1 depicts the software layers for untrusted subject clients. Note that for untrusted subjects, the actual TCB interface is the Windows NT operating system. However, the higher level client/server interfaces (i.e., SQL and Java interfaces) are useful for analysis, because the access control policy is defined on objects that are manipulated through these higher-level interfaces.

Several software components are outside the TCB on untrusted clients. Requests from these components are translated into SQL statements inside the CmdSeq protocol requests on the client side before being sent to the ASA server. The security of ASA does not depend on their correct operation.

Although **dbisqlc** is inside the TCB when used by a Database Administrator (DBA), it is outside the TCB when used by untrusted subjects. Untrusted subjects could bypass these layers by composing and sending requests across a Windows NT Named Pipe from an application program. However for untrusted subjects, the security of ASA does not depend upon the correctness of the higher levels of the TCB interface on the client side, and these layers can be safely bypassed, because SQL commands are passed to ASA for parsing and processing. All access control checks based on SQL statements, commands, stored procedures, and other higher level interfaces are made inside the ASA server.

4.2.3 Trusted Subjects

The middle of Figure 4.1 depicts the layers of a trusted client. Note that the TCB interfaces for trusted clients are DBISQL, database utilities, and ESQL. The communications layers include the DBLIB functions, the Command Sequence protocol and streams.

Trusted administrators (i.e., DBAs) need a trusted interface to perform trusted operations such as assigning passwords or backing up the audit log. When it is used by a DBA, **dbisqlc** is inside the TCB. However, **dbisqlc** is not policy enforcing, since SQL commands are sent to the Server for processing and all DAC checks are performed by the Server.

The following TCB interfaces are available to a user with DBA authority to administer the database:

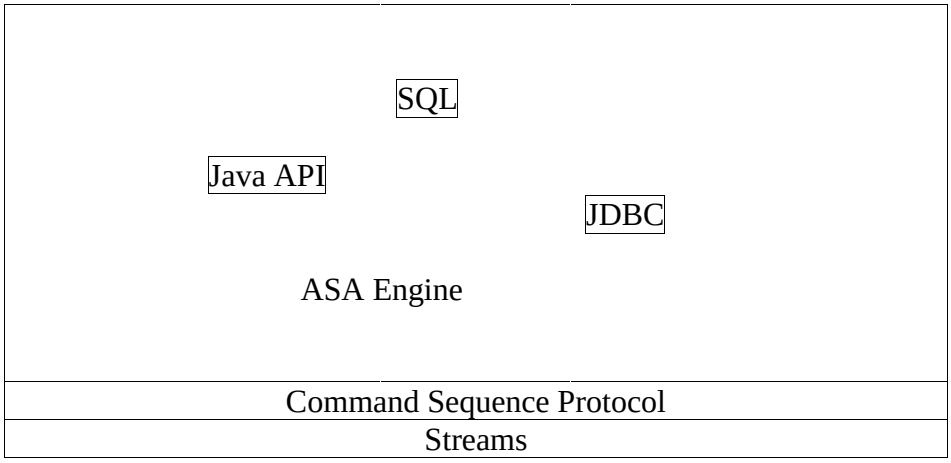
- **dbisqlc**,
- Database utilities, and
- ESQL

dbisqlc is an interactive utility that allows a user to execute SQL statements. When **dbisqlc** is used by a user with DBA authority, it is inside the TCB. The database utilities may be executed through a command line interface. The database utilities are also accessible through other interfaces including ESQL, function calls, and **dbisqlc**. SQL statements and database utilities can also be invoked through the ESQL programmatic interface.

4.2.4 The Server

The right side of Figure 4.1 depicts the server. All of the ASA software executing in a database server is inside the TCB. The server includes the streams and CmdSeq communications layers. The database engine also support internal TCB interfaces for SQL, the Java API, and JDBC. These internal interfaces are shown in Figure 4.2. Although these are not TCB interfaces from a software perspective, these are logical TCB interfaces. They are the interfaces that access database objects and the level at which a security policy must be defined. Logical TCB interfaces are described in Section 4.7, Logical TCB Interfaces.

Figure 4.2 - Internal ASA Engine Interfaces



4.3 Relational Database Objects

ASA is a relational database management system. Relational databases store data in tables organized into columns and rows. A column represents a single attribute such as name, color, or price. Each row stands for one instance of the topic or one record. Basic operations from the relational model are as follows:

- **Selection:** Retrieves a subset of rows in a table or tables. The selection criteria are specified in a WHERE clause. ASA retrieves only rows that meet the criteria.
- **Projection:** Retrieves a subset of the columns. A subset of columns is specified in the SELECT clause
- **Join:** Concatenates related information from two or more tables.

Data is accessed through a query language (SQL). The following basic operations on tables are called Data Manipulation Language (DML) Statements in SQL:

- **SELECT**: Retrieves data from a table or tables;
- **INSERT:** Adds a row to a table;
- **UPDATE**: Update values in a row; and
- **DELETE**: Removes a row from a table.

Data Definition Language (DDL) Statements in SQL are used to define SQL objects and their attributes. SQL objects are used to store ASA internal data such as User IDs, permissions, and file locations as well as user data. The ASA Named Objects are a subset of the SQL objects. Three basic DDL statements are:

- **CREATE:** Used to create SQL objects such as tables, views, procedures, triggers;
- **DROP:** Used to delete objects; and
- **ALTER:** Only the definitions of some objects such as tables can be altered. Others such as procedures have to be dropped and recreated.

DDL statements are used to GRANT and REVOKE authorities (such as DBA and Resource) and object permissions and to set database options.

SQL objects related to tables are indexes, integrity contraints, triggers, and views. Indexes are used to enhance performance. They are considered an attribute of tables.

Integrity of its data is critical for a Database Management System. ASA supports the following integrity mechanisms:

- Integrity constraints,
- Triggers, and
- Transactions.

Integrity constraints are considered to be table attributes and include the following:

- **Check conditions:** Specified on columns. Define a range of acceptable values.
- **Entity integrity constraints:** Specified on columns. Examples are non-null and unique.
- **Referential integrity contraints:** Specified between columns. Specifies that a foreign key in one table match the primary key in another or the same table.

A trigger is a set of SQL statements that can operate on the same or other tables. Triggers can be defined to fire on an INSERT, UPDATE, or DELETE operation on a table. Triggers execute with the security attributes of the owner of the table on which they are defined.

Transactions maintain integrity when multiple users are updating a database table simultaneously. Transaction integrity is maintained by locks that do not persist after a transaction is completed. (See Section 4.15, Transaction Processing for more information.)

Another type of SQL object is a view. Views are logical tables derived from other tables; they do not exist in physical form. Views are created through a stored SQL statement that performs selections, projections and joins. Views are named objects in ASA. (See Sections 5.3.1, Named Objects, for more information on named objects in ASA and the relationship between SQL objects and named objects.)

Procedures and functions are a set of SQL statements. They can include program control statements such as conditional clauses and case statements. Parameters may also be passed in and out of a procedure or function. A function is a specific type of procedure that returns a value. Procedures and functions are used for both performance and security. They can be compiled once and executed several times on the server. Procedures and Functions can be used to limit user access to tables to a predefined set of operations. Procedures are named objects in ASA and users can be granted *EXECUTE* permission on a procedure.

Relational database management systems are self-referencing. SQL object definitions are themselves stored in tables. The tables that store the definitions and attributes of SQL objects are called System Tables. The ASA System Tables are described in the next section.

4.4 System Tables

System tables are used to store SQL object definitions and attributes. Table 4.1 lists the system tables in ASA. Some important object related system tables are as follows:

- **SYSTABLE:** Stores table and view definitions. SYSTABLE is the most important system table since it is used to store the definitions of all the other system tables as well as tables and views created by users.
- **SYSCOLUMN:** Stores column definitions.
- **SYSPROCEDURE:** Stores procedure and function definitions.
- **SYSTRIGGER:** Stores trigger definitions.

For procedural SQL objects such as procedures, functions, triggers, and views, the text of the SQL statements is stored as a character string in the appropriate system table.

Users and their permissions are also stored in system tables. Users are SQL objects, but not named objects in ASA. Some important system tables related to users and permissions are as follows:

- **SYSUSERPERM:** Stores Usernames, User IDs, and authorities. Two authorities are used in the ASA evaluated configuration: DBA and Resource. The only trusted user in ASA is a user with DBA authority. Resource Authority gives a user the ability to create objects in a database.
- **SYSLOGIN:** Maps NT accounts to ASA User IDs and is used for integrated login.
- **SYSGROUP:** Stores group memberships.
- **SYSTABLEPERM:** Stores table and view permissions such as *SELECT, INSERT, UPDATE,* and *DELETE*.
- **SYSPROCPERM:** Stores *EXECUTE* permissions on procedures and functions.

Table 4.1 - System Tables

System Table	Description
DUMMY	Used to extract information from the database
SYSARTICLE	Describes articles in a SQL Remote publication.
SYSARTICLECOL	Identifies a column in an article.
SYSCAPABILITY	Contains information on the capabilities of remote servers.

SYSCAPABILITY_NAME	Contains names for each capability id SYSCAPABILITY
SYSCOLLATION	Contains the collation sequences available to ASA
SYSCOLLATIONMAPPINGS	Contains mapping label and collation sort order for collation sequence
SYSCOLPERM	Contains columns with UPDATE permission
SYSCOLUMN	Contains each column in every table or view
SYSDOMAIN	Shows the association between predefined data types (domains) and unique numbers assigned to them
SYSEVENT	Contains events.
SYSEVENTTYPE	Contains event types.
SYSEXTENT	(not used by ASA)
SYSEXTERNLOGINS	Contains alternate user names and passwords when connecting to a remote server.
SYSFILE	Contains names operating system files where data and logs are stored.
SYSFKCOL	Identifies the columns in the foreign key and associates each column in the foreign key with a column in the primary key of the primary table
SYSFOREIGNKEY	Contains general information about the foreign key
SYSGROUP	Describes many-to-many relationship between groups and members
SYSINDEX	Describes each index in the database
SYSINFO	Indicates the database characteristics, as defined when database was created using **dbinit.**
SYSIXCOL	Describes each column in the index
SYSJAR	Contains information about JAR files that store java packages.
SYSJARCOMPONENT	Contains information about Java component
SYSJAVACLASS	Contains all information related to Java classes
SYSLOGIN	Contains all the User Profile names that can be used to connect to the database using an integrated login
SYSOPTION	Contains option settings for the users
SYSPROCEDURE	Contains procedures defined in the database
SYSPROCPARM	Contains parameters to a procedure in the database
SYSPROCPERM	Describes many-to-many relationship between procedures and users who have permission to call these procedures
SYSPUBLICATION	Describes a SQL remote publication
SYSREMOTETYPE	Contains information about SQL Remote
SYSREMOTEUSER	Contains user IDs with REMOTE permissions (a subscriber), together with the status of SQL Remote messages that were sent to and from that user
SYSREMOTEOPTION	Contains the values of options of remote servers.
SYSREMOTEOPTIONTYPE	Contains information on options of remote servers.
SYSSERVERS	Used to list remote servers.
SYSSQLSERVERTYPE	Contains information relating to compatibility with Adaptive Server Enterprise
SYSSUBSCRIPTION	Describes a subscription from one user ID (which must have REMOTE permissions) to one publication
SYSSYNC	Contains information relating to MobilLink
SYSTABLE	Describes tables or views in the database
SYSTABLEPERM	Contains permissions given by one user to other users using GRANT command
SYSTRIGGER	Contains triggers that are automatically created by the database for foreign key definitions which have a referential triggered action
SYSTYPEMAP	Contains compatibility mapping values for the SYSSQLSERVERTYPE system table
SYSUSERMESSAGES	Contains user-defined message for an error condition
SYSUSERPERM	Contains description of user IDs such as user group, password, user name etc.

SYSUSERTYPE	Contains description of user-defined data types

The system tables contain all the information about the database. The system tables are owned by the group SYS. By default, most of the system tables are publicly readable, because the group PUBLIC is by default a member of the group SYS and all users are members of the group PUBLIC. This can be changed by removing PUBLIC as a member of the group SYS. The exceptions are that users can not access SYSUSERPERM and SYSUSERAUTH system tables that contain database-level permissions and password for each user ID. The contents of these tables cannot be modified using UPDATE, DELETE and INSERT commands, and the structure of the system tables cannot be changed using the ALTER TABLE and DROP commands.

It is not possible to connect to the database as user SYS. (SYS is created with a NULL password, which disables logins, and there is a check in the code to prevent the password from being changed to another value.)

System tables are stored in a database file on disk. They persist when the server is not running. When the server is running, objects referenced by a request are copied into object structures in the cache.

4.5 Tables and Views

This section provides a brief overview of tables and views from an architectural perspective introducing the system tables that are used to store table and view definitions. (Operations and permissions are discussed in Section 5.3, Objects and Section 6.2, Discretionary Access Control.)

4.5.1 Tables and Views

Tables are the basic objects of a relational database. Both system tables and user-defined tables are defined in the SYSTABLE system table as shown in Table 4.2. Note that in the case of a view, the text of the SELECT statement that defines the view is stored as a character string in the view_def column.

Table 4.2 - SYSTABLE System Table

Column name	Column type	Column constraint	Table constraints
table_id	UNSIGNED INT	NOT NULL	Primary key
file_id	UNSIGNED INT	NOT NULL	Foreign key references SYSFILE
count	INTEGER	NOT NULL	
first_page	INTEGER	NOT NULL	
last_page	INTEGER	NOT NULL	
primary_root	INTEGER	NOT NULL	
creator	UNSIGNED INT	NOT NULL	Unique index. Foreign key references SYSUSERPERM.user_id
table_name	CHAR(128)	NOT NULL	Unique index
table_type	CHAR(10)	NOT NULL	
view_def	LONG VARCHAR		

remarks	LONG VARCHAR		
replicate	CHAR(1)	NOT NULL	
existing_obj	CHAR(1)		
remote_location	LONG VARCHAR		
remote_objtype	CHAR(1)		
srvid	INTEGER		Foreign key references SYSSERVERS

4.5.2 Columns

The attributes of each column in a table are defined by a row in the SYSCOLUMN system table as shown in Table 4.3.

Table 4.3 - SYSCOLUMN System Table

Column name	Column type	Column constraint	Table constraints
table_id	UNSIGNED INT	NOT NULL	Primary key, foreign key references SYSTABLE.table_id
column_id	UNSIGNED INT	NOT NULL	Primary key
pkey	CHAR(1)	NOT NULL	
domain_id	SMALLINT	NOT NULL	foreign key references SYSDOMAIN.domain_id
nulls	CHAR(1)	NOT NULL	
width	SMALLINT	NOT NULL	
scale	SMALLINT	NOT NULL	
estimate	INTEGER	NOT NULL	
max_identity	BIGINT	NOT NULL	
column_name	CHAR (128)	NOT NULL	
remarks	LONG VARCHAR		
"default"	LONG VARCHAR		
"check"	LONG VARCHAR		
user_type	SMALLINT		Foreign key references SYSUSERTYPE.type_id
format_str	CHAR(128)		
column_type	CHAR(1)	NOT NULL	
remote_name	VARCHAR(128)		
remote_type	UNSIGNED INT		

4.5.3 Indexes

Indexes allow quick lookup of information. The index relates each indexed column value to the physical location at which the row of data containing the indexed value is stored. Indexes are defined in the SYSINDEX system table as shown in Table 4.4.

Table 4.4 - SYSINDEX System Table

Column name	Column type	Column constraint	Table constraints
table_id	UNSIGNED INT	NOT NULL	Primary key, Unique index. Foreign key references SYSTABLE.
index_id	UNSIGNED INT	NOT NULL	Primary key
root	INTEGER	NOT NULL	
file_id	SMALLINT	NOT NULL	
"unique"	CHAR(1)	NOT NULL	
creator	UNSIGNED INT	NOT NULL	Foreign key references SYSUSERPERM.
user_id	index_name	CHAR(128)	NOT NULL
Unique index	remarks	LONG, VARCHAR	

4.5.4 Foreign Keys

Foreign Keys are used to define referential integrity constraints. A foreign key is a relationship between two tables—the foreign table and the primary table. Every foreign key is defined by one row in SYSFOREIGNKEY as shown in Table 4.5 and one or more rows in SYSFKCOL as shown in Table 4.6. SYSFOREIGNKEY contains general information about the foreign key while SYSFKCOL identifies the columns in the foreign key and associates each column in the foreign key with a column in the primary key of the primary table.

Table 4.5 - SYSFOREIGNKEY System Table

Column name	Column type	Column constraint	Table constraints
foreign_table_id	UNSIGNED INT	NOT NULL	Primary key, foreign key references SYSTABLE.table_id. Unique index.
foreign_key_id	SMALLINT	NOT NULL	Primary key
primary_table_id	UNSIGNED INT	NOT NULL	Foreign key references SYSTABLE.table_id.
root	INTEGER	NOT NULL	
check_on_commit	CHAR(1)	NOT NULL	
nulls	CHAR(1)	NOT NULL	
role	CHAR(128)	NOT NULL	Unique index
remarks	LONG VARCHAR		

Table 4.6 - SYSFKCOL System Table

Column name	Column type	Column constraint	Table constraints
foreign_table_id	UNSIGNED INT	NOT NULL	Primary key. Foreign key references

			SYSCOLUMN.table_id. Foreign key references SYSFOREIGNKEY
foreign_key_id	SMALLINT	NOT NULL	Primary key. Foreign key references SYSFOREIGNKEY. foreign_key_id
foreign_column_id	UNSIGNED INT	NOT NULL	Primary key, Foreign key references SYSCOLUMN column_id
primary_column_id	UNSIGNED INT	NOT NULL	

4.5.5 Triggers

A trigger is a set of SQL statements associated with a table that executes automatically when a specific INSERT, UPDATE, or DELETE statement is executed on the table. Triggers may be used to enforce a referential integrity policy on a database. In addition, triggers allow changes to cascade through related tables, to aid in enforcing complex column restrictions, or to perform actions in response to data modifications. Triggers are created with the CREATE TRIGGER statement.

Each trigger in the database is described by one row in the SYSTRIGGER system table as shown in Table 4.7. The table also contains triggers that are automatically created by the database for foreign key definitions that have a referential triggered action (such as ON DELETE CASCADE).

Table 4.7 - SYSTRIGGER System Table

Column Name	Column Type	Column Description/Constraints
trigger_id	smallint	Each trigger is assigned a unique number (the trigger number), which is the primary key for SYSTRIGGER.
table_id	smallint	The table number uniquely identifies the table to which this trigger belongs. Foreign key references SYS.SYSTABLE table_id.
event	char(1)	The event or events that cause the trigger to fire. This single-character value corresponds to the trigger event that was specified when the trigger was created.
trigger_time	char (1)	The time at which the trigger will fire. This single-character value corresponds to the trigger time that was specified when the trigger was created: A - After, B - Before.
trigger_order	smallint	The order in which the trigger will fire. This determines the order in which triggers are fired when there are triggers of the same type (INSERT, UPDATE or DELETE) that fire at the same time (before or after).
foreign_table_id	smallint	The table number of the table containing a foreign key definition which has a referential triggered action (such as ON DELETE CASCADE). Foreign key references SYS.SYSFOREIGNKEY
foreign_key_id	smallint	The foreign key number of the foreign key for the table referenced by foreign_table_id. Foreign key references SYS.SYSFOREIGNKEY
referential_action	char(1)	The action defined by a foreign key. This single-character value corresponds to the action that was specified when the foreign key was created: (C - CASCADE; D - SET DEFAULT; N - SET NULL; R - RESTRICT)

trigger_name	char (128)	The name of the trigger. One table cannot have two triggers with the same name.
trigger_defn	long varchar	The command that was used to create the trigger.
remarks	long varchar	A comment string.

4.6 Procedures and Functions

Procedures and functions are a series of Structured Query Language (SQL) statements to be executed as a unit. Procedures and functions may be used to group a set of SQL statements for the convenience of users or to provide a controlled means of granting access to tables. Procedures and functions may be either system or user-defined. System stored procedures are installed with the system and the interfaces are described in Section 4.7.2, Stored Procedures Interfaces. System functions are described in Section 4.7.1, SQL.

Procedures are created with the CREATE PROCEDURE statement and deleted with the DROP PROCEDURE statement. Procedures are executed using the CALL statement. A Function is a procedure that returns a value. User-defined functions are created with the CREATE FUNCTION statement and deleted with the DROP FUNCTION statement. ASA user-defined functions are called in the same way as SQL functions using a SELECT statement. Procedures can access tables and views, and can call other procedures.

From an architecture and access control perspective, procedures and functions are handled the same way. When a procedure or function is defined, its definition and the definitions of its parameters are stored in the SYSPROCEDURE and SYSPROCPARM system tables, respectively. When a procedure or function is called, its definition is loaded into the procedure definition structure in the ASA heap section of the cache. The text of the procedure definition is compiled into executable form and stored.

Procedures and functions are owned by their creators. DBA authority is required to create a procedure or function for another user. The owner of a procedure or function is stored in the SYSPROCEDURE table. Owners can execute their own procedures and functions. A procedure executes with the permissions of its owner, not the calling user.

The only permission on procedures and functions is *EXECUTE*. *EXECUTE* permissions are stored in the SYSPROCPERM system table. Each row of the SYSPROCPERM table corresponds to one user granted *EXECUTE* permission on one procedure. When a user is connected, their *EXECUTE* permissions on procedures and functions are stored in a permission list that is pointed to by the user's connection structure. The DAC check performed when a procedure or function is executed is an auditable event.

Table 4.8 defines the SYSPROCEDURE system table. Each procedure or function has a unique proc_id that serves as the primary key for the table. Each procedure can also be uniquely identified by its creator and procedure name, since one creator cannot have two procedures with the same name. The procedure definition column contains the text of the command used to create the procedure including the set of SQL statements in ASCII format to be executed by the procedure or function.

Table 4.8 - SYSPROCEDURE System Table

Column Name	Column Type	Column Description/Constraints
proc_id	smallint	Each procedure is assigned a unique number as its primary key.
creator	smallint	Foreign key references SYS.SYSUSERPERM user_id.
proc_name	char (128)	The name of the procedure. One creator cannot have two procedures with the same name.
proc_defn	long varchar	The command that was used to create the procedure.
remarks	long varchar	A comment string.
replicate	char (1)	(Y/N) Indicates whether the procedure is a primary data source in a Replication Server installation. Not applicable in the evaluated configuration.

Table 4.9 describes the SYSPROCPARM system table. SYSPROCPARM contains one row for each parameter of the procedure or function. The primary key is composed of the procedure ID and the parameter ID. The other columns define the data type and whether the parameter is an input parameter or an output parameter.

Table 4.9 - SYSPROCPARM System Table

Column Name	Column Type	Column Description/Constraints
proc_id	smallint	Uniquely identifies the procedure to which this parameter belongs. Primary key, foreign key references SYS.SYSPROCEDURE
parm_id	smallint	Each procedure starts numbering parameters at 1. The order of parameter numbers corresponds to the order in which they were defined. Primary key.
parm_type	smallint	The type of parameter will be one of the following: Normal parameter (variable) Result variable - used with a procedure that returns result sets SQLSTATE error value SQLCODE error value
parm_mode_in	char(1)	(Y/N) Indicates whether this parameter supplies a value to the procedure (IN or INOUT parameters).
parm_mode_out	char(1)	(Y/N) Indicates whether this parameter returns a value from the procedure (OUT or INOUT parameters).
domain_id	smallint	Identifies the data type for the parameter, by the data type number listed in the SYSDOMAIN table. Foreign key references SYS.SYSDOMAIN.
width	smallint	Contains the length of a string parameter, the precision of a numeric parameter, or the number of bytes of storage for any other data types.
scale	smallint	The number of digits after the decimal point for numeric data type parameters, and zero for all other data types.
parm_name	char(128)	The name of the procedure parameter.
remarks	long varchar	A comment string.
default	long varchar	The default value for the parameter, held as a string.
user_type	integer	The user type of the parameter.

Table 4.10 describes the SYSPROCPERM system table. SYSPROCPERM contains one row for each user granted *EXECUTE* access on a procedure or function. The primary key is composed of

the procedure ID and the user_id of the grantee. Since the only permission that can be granted on procedures and functions is *EXECUTE*, no other information is required**.**

Table 4.10 - SYSPROCPERM System Table

Column Name	Column Type	Column Description/Constraints
proc_id	smallint	Primary key. The procedure number uniquely identifies the procedure for which permission has been granted.
grantee	smallint	The user number of the user ID receiving the permission. Primary key, foreign key references SYS.SYSUSERPERM user_id.

4.7 Logical TCB Interfaces

The logical interfaces to the ASA TCB are:

- SQL that is generated by the DBISQL and the ESQL interfaces,
- Stored Procedures,
- Java Application Programming Interface (API), and
- Database utilities.

4.7.1 SQL

SQL statements may be sent through the Database Interactive SQL (DBISQL) interface or the Embedded SQL (ESQL) programming interface. There are four components to the SQL interface:

- SQL Statements,
- Built-in SQL Functions,
- SQL Global Variables, and
- System stored procedures.

SQL functions and global variables are accessed through the SELECT statement. System Procedures are accessed through a CALL statement. System stored procedures are part of the SQL interface, but are discussed separately in Section 4.7.2 due to their complexity.

4.7.1.1 SQL Statements

SQL statements typically consist of a verb followed by a SQL object followed by one or more modifying clauses. Named objects are a subset of the objects referenced by SQL statements. The SQL statements are summarized in a table in Appendix C that includes the permissions required to execute each statement.

4.7.1.2 Built-in SQL Functions

The SQL interface also includes built-in SQL functions that can be accessed using a SELECT command. SQL provides the following types of built-in functions:

- **Aggregate** functions such as count, maximum, or sum,
- **Numeric** functions such as cosine, exponent, or round,
- **String** functions: char_length or substring
- The **Text and image** function is textptr, which returns the 16-byte binary pointer to the first page of the specified text column.
- **Date and time** functions such as hour and year,
- **Data type conversion** functions such as integer to hex or expression to string,
- **System** functions, and
- **Miscellaneous** functions such as isnull or return plan.

The only SQL functions that are security relevant are the system functions that provide interfaces to read server properties, database properties, and connection properties. There are three SQL system functions to retrieve properties:

- CONNECTION_PROPERTY: Returns the specified connection property as a string.
- DB_PROPERTY: Returns the value of the specified database property as a string.
- PROPERTY: Returns the value of the specified server-level property as a string.

Some properties map to security relevant configuration options such as the minimum password length and login mode. However, these functions can only read the values of these public objects, not set them.

4.7.1.3 Global Variables

SQL also provides a global variable interface that allows users to retrieve values set by ASA. The global variables are described in Table 4.11 below. Most of them are not security relevant as they are local to a connection as indicated by the last column. The other three Identity, Servername, and Version are public objects.

Table 4.11 - SQL Global Variables

Variable	Description	Local to a Connection
Error	Error status	Yes
Identity	Last value inserted into an identity column	No
Isolation	Current isolation level of active connection	Yes
Procid	Stored procedure ID of currently executing process	Yes
Rowcount	Number of rows affected by last statement	Yes
Servername	Name of current database server	No
Sqlstatus	Status information from last fetch	Yes
Version	ASA version number	No

4.7.2 Stored Procedures Interfaces

ASA provides system and catalog stored procedures to carry out administrative tasks. System procedures are built-in stored procedures used for getting reports from and updating system tables.

Catalog stored procedures retrieve information from the system tables in tabular form. Examples of system stored procedures are as follows:

- **sa_validate**: Validates all tables in the database.
- **sa_table_page_usage:** Reports information about the usage of database tables.
- **sp_login_environment**: Sets the ODBC and Embedded SQL connection options.
- **sp_tsql_environment**: In the evaluated configuration, this stored procedure does nothing except raise an error condition.

Catalog stored procedures return sets displaying server, database, and connection properties in tabular form. Examples of these include the following:

- **sa_conn_info**: Returns a subset of the connection properties.
- **sa_conn_properties:** Returns values for each available connection property
- **sa_db_info**: Returns a single row containing the Number, Alias, File, ConnCount, PageSize, and LogName for the specified database.
- **sa_db_properties:** Returns database ID number, and the Number, PropNum, PropName, PropDescription, and Value for each property returned by sa_db_info.
- **sa_eng_properties:** Returns the value of each available engine property.

4.7.2.1 Ownership and Privilege

Stored procedures belong to a database and are created when a database is created. When a database is created in the evaluated configuration, 206 stored procedures and functions are created as part of the database creation process. As the table in Appendix D shows, the vast majority of these are owned by DBO, and have execute permission granted to PUBLIC. The group DBO has DBA authority. Stored procedures execute with the permissions and privilege of the owner of the procedure. (See Section 4.7.2.2.2, Stored Procedures that Check Permissions.) Two of the stored procedures are owned by SYS. The other stored procedures are owned by a special group, rs_systabgroup.

The evaluated configuration is installed with a demonstration database called asademo.db. This database has 22 additional stored procedures, for a total of 237 stored procedures. The additional ones include stored procedures for the Customer Database demonstration, as well as JDBC stored procedures discussed below.

4.7.2.2 Types of Stored Procedures

During its analysis of the TCB interface provided by stored procedures, the team divided the stored procedures into the following types. These types correspond to the codes in the comments field of the table in Appendix D. Appendix D contains a complete list of all the system and catalog stored procedures in ASA, and theirs types as described in the subsections that follow, including whether or not DBA or Resource authority is required to execute them.

4.7.2.2.1 Stored Procedures for Compatibility

These stored procedures, identified as type "1" in the table in Appendix D, make up over one half of the first 100 stored procedures. These stored procedures call "sp_tsql_feature_not_supported", which declares an exception "feature_not_supported" and signals it. The stored procedure "sp_tsql_feature_not_supported" does nothing else. Those stored procedures identified as type 1 also do no other action besides calling "sp_tsql_feature_not_supported".

4.7.2.2.2 Stored Procedures that Check Permissions

These stored procedures, identified as types "2" or "3" in the table in Appendix D, call the stored procedure "sp_checkperms" as their first action. The stored procedure "sp_checkperms" then checks the input variable for the value "DBA" if type 2 or "Resource" if type 3. If input is not either, then sp_checkperms signals permission denied. The input value is passed as text, not as a user id or user identifier. If the check passes, the rest of the calling stored procedure is executed. Therefore, even though PUBLIC can start execution of most stored procedures, this additional check means that only a user with DBA authority can successfully execute some stored procedures, and only a user with Resource authority can successfully execute others.

4.7.2.2.3 Stored Procedures that Call Another Stored Procedure

These stored procedures, identified as type "5" in the table in Appendix D, call another stored procedure (that is not sp_checkperms or sp_tsql_feature_not_supported.) For example, proc_id 213 calls sp_jconnect_trimit to get desired information. The stored procedures may be called with the "call" or "execute" syntax.

4.7.2.2.4 Internal Procedures

These stored procedures, identified as type "6" in the table in Appendix D, call an internal procedure (which can be thought of as a database engine entry point). Internal procedures are procedures that do not use SQL to perform their functions. Instead, these procedures call compiled functions inside the main ASA executable (dbsrv7.exe or dbeng7.exe). Internal procedures can only be created by Sybase as they require implementation in C/C++ inside the server. (I.e., new internal procedures cannot be added to ASA 7.0.0 with C2 Update, since the code has already been frozen and the software released.) Many of the stored procedures with "java" in the stored procedure name use this syntax. Permission checks for these stored procedures are done as for any stored procedure.

In the evaluated configuration, a user cannot add new internal procedures, for two reasons:

- Only SYS can create stored procedures using the internal name syntax, and it is not possible to connect to any database as user SYS in the evaluated configuration. During creation or upgrading of a database, the database engine fakes a connection as user SYS to be able to create the two stored procedures owned by SYS, as well as to be able to add and alter system tables.

- The engine has a table listing all of the internal procedures, and the C routine to which they correspond. To add a new internal procedure, this table must be changed, a new procedure added, and the engine recompiled to add code to dbserv7.dll to implement it.

Examples of the internal procedures called include the following:

- ‘table_page_usage’
- ‘sa_proc_debug_connect’
- ‘sa_proc_debug_attach_to_connection’
- ‘sa_exec_script’ (proc_id 169; accepts a filename as input)
- ‘sp_forward_to’ (using the at anyserver syntax, proc_id 178)
- 'index_levels'
- 'sa_app_get_status'
- 'sa_app_set_infoStr'
- 'sa_app_get_infoStr'
- 'sa_conn_set_status'

The following four internal stored procedures are executable only by a user with DBA authority:

- sa_table_page_usage: puts information about page usage into a table
- sa_flush_cache: writes all dirty pages in the cache to disk
- sa_exec_script: executes a SQL script file (only used by Sybase Central which is not in the evaluated configuration)
- sa_server_option: sets the value of a server option ("request_level_debugging", "disable_connections", "liveness_timeout", or "quitting_time")

The following internal stored procedure are used for Java debugging and cannot be used in the evaluated configuration. The ten internally defined Java debugging procedures are all executable by PUBLIC. It is not possible to do Java debugging in the evaluated configuration because Java debugging requires networking with TDS and TCP/IP, which are not in the evaluated configuration. These are the stored procedures for Java debugging:

- java_debug_attach_to_vm(in vm_name char(128),out debugger long binary)
- java_debug_connect(in user_to_debug char(128))
- java_debug_detach_from_vm(in debugger long binary)
- java_debug_disconnect()
- java_debug_free_existing_vms()
- java_debug_get_existing_vms(out buffer long binary)
- java_debug_get_vm_name(in vm_handle long binary,out vm_name char(128))
- java_debug_release_vm(in vm_handle long binary)
- java_debug_request(in debugger long binary,in request long binary,out out_request long binary)
- java_debug_wait_for_debuggable_vm(out buffer long binary)

4.7.2.2.5 Extended Procedures

These stored procedures, identified as type “4” in the table in Appendix D, call an external entry point that is a compiled function in a dynamic link library outside the main ASA executable. These functions, as part of a dynamic link library used by ASA, share the ASA process space. The syntax allows identification of Windows 3.x, Windows 95, Windows NT, OS/2, UNIX, or Netware executables. The syntax allows calling external Dynamic Link Library (DLL) from a stored procedure or user-defined function when the operating system supports DLLs.

Only a DBA can install an additional extended procedure, and the TFM specifically prohibits this action in the evaluated configuration. These additional extended procedures must use existing DLL functions with defined external entry points. In the evaluated configuration, a user cannot add new external entry points. This would require addition of a new DLL, or recompiling an existing DLL.

Several extended procedures are briefly described here. **xp_cmdshell** allows a console-interface command to be executed by the operating system. For example, an operating system command that erases files could be executed through this extended stored procedure. Only a user with DBA authority can execute **xp_cmdshell** by default.

Three extended stored procedures (**xp_startmail, xp_stopmail,** and **xp_sendmail**) implement Microsoft's Mail Application Programming Interface (MAPI). With these a user can send and receive mail from within the database. Only a user with DBA authority can execute these by default.

The following extended procedures are not security relevant:

- **xp_msver** returns product, company, version, and other information about ASA. It is not executable except by a user with DBA authority.

- **xp_sprintf** builds up a string from a format string and a set of input strings. It is executable by PUBLIC.

- **xp_scanf** extracts substrings from an input string and a format string. It is executable by PUBLIC.

The extended stored procedures **xp_read_file** and **xp_write_file** require DBA authority to execute. The TFM directs the DBA not to grant EXECUTE permission on these stored procedures.

4.7.2.2.6 Other Stored Procedures

The stored procedures that are not identified by type in the table in Appendix D either return an output parameter or return a result. Some of these stored procedures are used to get information from system tables or tables owned by the database owner. Others of these stored procedures get information from demonstration databases. Two of these are for use by SYS and are not granted to any other user. The majority are created by DBO, but granted to Public.

4.7.2.2.7 JDBC-related Stored Procedures

There are nineteen JDBC related stored procedures. In the evaluated configuration *EXECUTE* permission is granted to PUBLIC for these procedures. They are security-relevant, because JDBC allows the user to pass SQL statements from a Java class to the ASA query processing engine. The SQL statements passed in this manner are parsed and their permissions are checked by the server exactly as if they were received through a client interface for SQL like dbisqlc.exe.

4.7.3 Java Application Programming Interfaces (APIs)

Java classes are stored in ASA databases as data types that can be used as the data type of a column in a table or a variable. Java class instantiations (objects) can be saved as a value in an ASA table cell. Java objects can be inserted into a table, SQL SELECT statements can be executed against the fields and methods of the Java objects stored in a table, and Java objects can be retrieved from the table.

The ASA runtime Java classes (i.e., the minimum set of classes necessary to run ASA Java applications) present a logical TCB interface by providing an Application Programming Interface (API) to write stored procedures in Java to access relational data stored in ASA databases.

4.7.3.1 ASA runtime Java classes

The ASA runtime Java classes are organized into sets of classes. These sets of classes are called packages. The ASA runtime Java packages are as follows:

- sun
- java
- java.sql
- com.sybase
- sybase.sql.asa

The 'sun' and 'java' packages are licensed from Sun Microsystems and contain the Sun Microsystems Java runtime classes.

'java.sql' (a.k.a. JDBC) is the package that defines the API for connecting to ASA databases and executing SQL statements. JDBC is Sun's standardized Java API for writing database applications. ASA support for 'java.sql' is provided by 'com.sybase' and 'sybase.sql.asa'.

'com.sybase' implements the Sybase jConnect JDBC driver. However, Java applications do not have the ability to create and connect to sockets in the evaluated configuration. Therefore, the evaluated configuration does not support client-side JDBC. If sockets could be created and connected to in the evaluated configuration, jConnect would enable client-side JDBC support. This would provide the capability to connect to an ASA database using TCP/IP from a client application that is not running on the ASA server. Note that server-side JDBC does not use jConnect and does not rely upon 'com.sybase'. (See Section 4.7.3.1.2, JDBC Implementation for details.)

'sybase.sql.asa' implements server-side JDBC support by implementing 'java.sql' interfaces and extending 'java.sql' classes. The 'sybase.sql.asa' package also implements functionality that is accessible by non-administrative users that is not defined in the JDBC standard. However, the only publicly-accessible non-JDBC functionality exposed in this package are two Java methods from the 'sybase.sql.ASAUtils' package that support object serialization and de-serialization, 'toByteArray()' and 'fromByteArray()', as documented in the ASA User's Guide.

4.7.3.1.1 JDK implementation

The 'sun' and 'java' packages are licensed from Sun Microsystems and contain the Sun Microsystems Java runtime classes. The ASA implementation in the evaluated configuration supports a subset of the runtime classes and class methods defined in version 1.1.6 of the Sun Microsystems Java Development Kit (JDK) standard. In other words, the complete set of JDK 1.1.6 runtime classes and class methods are not in the evaluated configuration. (The following entire JDK 1.1.6 packages that are not physically present in the evaluated configuration: java.applet, java.awt, sun.applet, sun.audio, sun.awt, sun.beans.editors, sun.beans.infos, sun.jdbc.odbc, sun.misc, sun.net, sun.rmi, sun.security, sun.tools, sunw.io, and sunw.util.)

java.net is the only JDK 1.1.6 networking-functionality-related package that is physically present[3] in the evaluated configuration, but it is disabled. Underlying ASA function tables were updated to disable datagram, multicast, and stream socket functionality in order to meet system architecture requirements. Disabling networking functionality results in individual methods behaving as if the underlying socket call failed (e.g., 'java.net.SocketError' is thrown when trying to open a socket), preventing the establishment of network connections.

The complete set of corresponding Java native methods is not physically present in the evaluated configuration for the following JDK 1.1.6 classes and invoking those native methods will result in an undefined link exception: java.lang.ClassLoader, java.lang.Compiler, java.lang.Runtime, java.lang.SecurityManager, java.lang.Thread, java.io.File, java.io.FileDescriptor, java.io.FileInputStream, java.io.FileOutputStream, java.io.ObjectInputStream, java.io.ObjectOutputStream, java.io.ObjectStreamClass, java.io.PrintStream, java.io.RandomAccessFile, java.math.BigInteger, java.net.PlainDatagramSocketImpl, java.util.zip.Adler32, java.util.zip.CRC32, java.util.zip.Deflater, and java.util.zip.Inflater.

4.7.3.1.2 JDBC implementation

'java.sql' (a.k.a. JDBC) is the package that defines the API for connecting to ASA databases and executing SQL statements. The ASA implementation of JDBC is compliant with version 1.1 of the Sun Microsystems JDBC standard. However, given that only server-side JDBC is supported in the evaluated configuration, only Java applications running on the ASA server can use the JDBC API access ASA databases.

The ASA Java implementation does not support sockets in the evaluated configuration, and because of this, only server-side JDBC is supported (jConnect relies upon TCP/IP).

Instead of using the TCP/IP-based jConnect driver to connect either locally or remotely to ASA databases, Java native methods (i.e., non-Java applications and/or libraries) are used by server-side JDBC applications to connect locally to the underlying ASA database. In either case, connections are established via the JDBC API using the 'DriverManager.getConnection' method. In the case where a server-side JDBC application is establishing a connection, the URL that is passed as an argument consists of a string with zero length (instead of a string containing the URL of an ASA

[3] The JDK 1.1.6 'java.net' package and its corresponding Java native methods are physically present in the evaluated configuration with the exception of the PlainDatagramSocketImpl class, which is not physically present in any ASA configuration.

database). This causes a native method to be invoked instead of jConnect, which establishes a connection to the database. The credentials that were used to establish the database connection in the first place are used to connect again to the database, this time using Java native methods.

4.7.3.2 ASA runtime Java native methods

While Java runtime classes initiate requests to and receive responses from the underlying DBMS (and operating system, through the DBMS), they do so indirectly using the Java Native Interface (JNI) mechanism. The JNI mechanism implemented in the ASA Java Virtual Machine (JVM) allows the invocation by Java applications of native methods (non-Java applications and/or libraries). Native methods pass requests to and receive responses from the ASA DBMS (and operating system, through the DBMS). Native methods make the actual requests to and receive the actual responses from the underlying DBMS (and operating system, through the DBMS). In other words, native methods are the entities in the ASA Java implementation that are performing the actual security-relevant events.

Native methods included in the evaluated configuration implement only that functionality that is defined by the logical TCB interface that the runtime classes present.

Note that the Java runtime classes, the JVM, and the native method implementations together present a single software TCB interface. (This is discussed in more detail in Section 4.16.4, ASA JVM TCB Interface.)

4.7.4 Database Utilities

ASA includes a set of utility programs for backing up databases and performing other database administration tasks. Table 4.12 provides a brief description of each of these utilities and their required permissions.

Table 4.12 - Database Utilities

Utility	Command Line	Program Interface, if available	Description	Permissions
Backup	**dbbackup**	DBBackup	Makes copy of database files	DBA authority
		DBTruncate Log	Truncates a transaction log.	DBA authority
Collation	**dbcollat**	DBCollate	Extracts a collation sequence from a database.	DBA authority
Compres-sion	**dbshrink**	DBCompress	Reads input database file and creates compressed database file.	DBA authority
Erase	**dberase**	DBErase	Erases a database file and/or a transaction log file.	DBA authority
Information	**dbinfo**	DBInfo	Returns information about a database such as date/time created, log file names, page size, and connection parameters.	Need to be DBA for -u (page usage) option; valid user ID otherwise.
	dbinfo -u	DBInfoDump	Returns output page usage statistics.	DBA authority

Initialization	**dbinit**	DBCreate	Creates a database.	Creates DBA user ID with password SQL
Interactive SQL	**dbisqlc**		Interactive user interface for submitting SQL commands.	Valid User ID
Locate	**dblocate**		Lists ASA servers on network running TCP/IP.	Not in evaluated configuration
Log Transfer	**dbltm**		Replication agent.	Not in evaluated configuration.
Log Translation	**dbtran**	DBTranslateLog	Translates a transaction log file to SQL.	DBA authority
Ping	**dbping**	DB_string_ping_server	Attempts to create a streams-level connection to an engine or server, optionally attempts database connection.	Valid User ID to use -d option, none otherwise.
Rebuild	**rebuild**		Allows rebuilding of database with new configuration options such as page size, tailing blanks, and case sensitivity	DBA authority
Service Creation	**dbsvc**		Creates, deletes, and modifies NT Services for ASA server processes.	NT Administrator
Stop	**dbstop**		Stops a database server.	Set by –gk server switch. DBA in evaluated configuration.
Transaction Log	**dblog**		Displays or changes the name of the transaction log or transaction log mirror. Starts or stops a transaction log or transaction log mirror.	DBA authority.
	dblog -t	DBChange LogName	Sets the name of the transaction log file.	
Uncompres- sion	**dbexpand**	DBExpand	Uncompresses a database file.	DBA authority.
Unload	**dbunload**	DBUnload	Unloads a database.	DBA authority.
Upgrade	**dbupgrad**	DBUpgrade	Converts a database file from an earlier version of ASA to ASA 7.0.0.	DBA authority.
Validation	**dbvalid**	DBValidate	Validates all or part (i.e., selected tables) of a database	DBA authority.
Write File	**dbwrite -d**	DBChange WriteFile	Changes a write file to refer to another database file.	DBA authority
	dbwrite -c	DBCreate WriteFile	Creates a write file	
	dbwrite -s	DBStatus WriteFile	Gets status of a write file.	
SQL Preproces-sor	**sqlpp**		Processes a C or C++ program with embedded SQL statements	None.

1

4.8 Communications

The following communications layers are inside the TCB: streams, Command Sequence protocol, and DBLIB.

- Streams is a set of routines that translates generic communication requests such as connect or send packet into requests that are specific to the OS communications mechanism being used. (I.e., Windows NT Named Pipes in the evaluated configuration). Streams is inside the TCB both on the server and for trusted clients.

- The Command Sequence (CmdSeq) protocol is a presentation layer protocol used to transfer requests and responses between a client and ASA. CmdSeq is inside the TCB both on the server and for trusted clients.

- DBLIB is a dynamic link library called by the administrative utilities on the client side. DBLIB is inside the TCB when called by trusted subjects. DBLIB is an intermediate level that is called by ISQL, database utilities and ESQL. DBLIB calls the CmdSeq protocol.

4.9 Database Files

An Adaptive Server Anywhere database is made up of the following types of files:

- Main Database file
- Dbspace Files
- Transaction Log file
- Transaction Log Mirror file
- Database Utility Audit Log file
- Write File, and
- Temporary File.

These files are protected by OS DAC as described in Section 7.1, System Architecture.

4.9.1 Main Database File

The main database file is created when a database is initialized using **dbinit** utility or the CREATE DATABASE SQL statement. The main initial database file contains the system tables, database tables, and indexes. It also contains the checkpoint log, the rollback log and columns statistics. The first page after the definition page contains the first page of the SYSTABLE system table. Since the SYSTABLE system table contains pointers to all of the other tables in the database, this gives ASA a starting point. The main database file is also called the root database file and is named <*dbname*>.db.

4.9.2 Dbspace Files

Additional database or dbspace files can be created by a user with DBA authority files using the CREATE DBSPACE command. A main database file can have a maximum of twelve dbspace files

associated with it. Dbspace files can reside on the same or a different device from the main database file. Dbspace files store tables and indexes like the main database file, but not system tables. DBSPACE files can be used to improve performance by placing tables or indexes that are frequently accessed together on different physical devices. The system table SYSFILE keeps a record of all the dbspace files associated with a main database file.

The IN option of the CREATE TABLE or CREATE INDEX command can be used to place a table or index in a specific database file, either the main database file or a dbspace file. By default, if a database file is not specified, a table is placed in the main database file and an index is placed in the same database file as its table. A table or index must fit within one database file; it may not span more than one file.

4.9.3 Transaction Log File

Transaction log records are stored in the transaction log file. A database must be created with the transaction log set to ON in the evaluated configuration. Every action that the database engine takes against the database is stored in the transaction log in the order in which it occurs. This includes inserts, updates, deletes, commits, rollbacks, and database schema changes. When recovering from system or media failure, the database engine uses the transaction log to recreate lost data. The transaction log is also used to store audit records. By default, the transaction log is placed in the same directory as the main database file. However, to protect against media failure corrupting both the transaction log and the main database file, the two files should reside on different physical devices. Placing the main database file and transaction log on separate physical devices can also improve performance. The transaction log file is named <*dbname*>.log.

4.9.4 Transaction Log Mirror File

A transaction log mirror is an identical copy of the transaction log. It provides extra protection for critical data. A transaction log mirror is maintained concurrently with the transaction log. Every time a database change is recorded in the transaction log, it is also written to the transaction log mirror. The transaction log mirror is named <*dbname*> .mlg.

4.9.5 Database Utility Audit Log File

The audit log file is used to store audit records generated by ASA database utilities when the engine is not running. The file is in ASCII text format. The following utilities can generate records in the Audit Log File: **dblog, dbtran**, and **dbwrite**. The audit log file is named <*dbname*>.alg.

4.9.6 Write File

A write file provides a method of updating a read-only database such as one on a read-only compact disk (CD). Changes made to the read-only database are saved in the associated write file. The write file is named <*dbname*>.wrt.

4.9.7 Temporary File

ASA uses the temporary file when extra space is needed for certain operations such as sorting and performing unions. There is one temporary file per database and it is deleted when the database is closed. By default, the temporary file is created in the same directory as the main database file. Temporary files have the default extension .tmp.

All files consist of pages. Pages can be up to 32K bytes. Page size is set when the database is created, and fixed for all files used by the database. The ASA page size need not be the same as the operating system page size.

Pages are identified by a 32-bit number. The first four bits are the file number and the last 28 bits are the offset in within the file. The four bit file number is associated with the file types described above as follows: main database file (0), dbspace files (1-12), write file (13), transaction log file (14), and temporary file (15).

4.10 Page Management

Pages may be stored on a file on disk or in memory in the cache or both. The types of pages are as follows: Definition, Table, Index, Transaction log, Rollback log, Lock, Statistics, Temporary File, and Free.

4.11 Memory Management

ASA uses memory for the following:

- Cache for quick access to frequently required information,
- Dynamic memory pool for managing database connections, and
- Communications buffer pool for network communications.

4.11.1 Cache

The cache is used to hold data that is accessed frequently, so that it does not have to be fetched from disk each time it is needed. Each time the database engine needs to access data, either that data is available in the cache, or it must be retrieved from the file on disk. The size of the database engine cache can grow dynamically. There are command line switches to set minimum/maximum values as well as to set the cache to a fixed size. The ASA virtual memory management system moves information between the cache and a temporary file on disk. The database engine cache is the memory pool used by the core database engine for the following purposes:

- Cache pages, and
- Engine stacks and data structures.

4.11.1.1 Cache Pages

The database engine divides its cache into pages of a fixed size for holding information from the database files, and for holding other information that can be moved into the temporary file. By default, when an engine is started with a single database, the cache page size matches that of the database. If more than one database is loaded when a server is started, the cache page size is set by default to the maximum of the database page sizes.

The virtual memory management system retrieves pages from disk into memory for use when they are needed, and writes them out to disk when they are no longer needed or when there is no more

cache available. While pages are in use by the engine, they are *locked*. Unlocked pages are managed using the 2Q algorithm, similar in purpose but more refined than the Least Recently Used (LRU) algorithm. The contents of the pages selected by the 2Q algorithm are either written to disk (if the page is dirty) or erased (if it is not dirty) and then the page is reused.

Cache pages are used for the following:

- Database pages,
- Temporary table pages,
- Heaps,
- Lock table,
- Column statistics registry, and
- Table page map.

4.11.1.1.1 Database table and index pages

Any time a database page is read from disk, it is read into a cache page. If changes are made to the database through an INSERT, UPDATE, DELETE or DDL statement, then the changes are made to the cache pages and these cache pages are marked as *dirty***.**

4.11.1.1.2 Temporary table pages

Temporary tables are built-in pages from the temporary file. These pages occupy cache pages when they are being used, and then are eliminated from the cache in the normal LRU fashion.

4.11.1.1.3 Heaps

All dynamic memory in the database engine is allocated out of *heaps*. A *heap* is a collection of cache pages that are used together. Sub-allocation is done within each heap. As all memory is managed out of cache pages, no piece of memory can be allocated that is larger than the size of a cache page. Any data structure in the engine that would require a large contiguous piece of memory has been modified to exist in pieces. There are two different kinds of heaps: the main heap and relocatable heaps.

There is one main heap that is never relocated or paged out to disk. It contains many different data structures, enumerated below. The main heap stores many of the engine data structures. The following data structures are stored in the main heap: database, connection, user descriptor, table and view description, column description, index description, statement handle, cursor handle, procedure handle, permissions, string ID table, and cursor positions. One data structure is used for each instance of each object. For example, a table with six columns requires one table description data structure plus six column description data structures.

Relocatable heaps are used for certain types of data structures that are well contained, such as statement parse trees. Relocatable heaps can be paged out to disk in the temporary file and relocated when reloaded. These heaps contain pointers within the heap itself or to the main heap. When a relocatable heap is reloaded and moved, the pointers within the heap are fixed up to represent the new page locations. The following objects are each allocated a relocatable heap:

- Parsed (prepared) statements including view definitions, table check constraints, procedure definitions and trigger definitions,
- Database engine cursors,
- Stored procedures and triggers,
- Stored procedure and trigger execution context, and
- Nested procedure and trigger invocations

4.11.1.1.4 Lock Table

The lock table stores all the row lock pages in a singly linked list. Lock table pages can be paged out to the temporary file.

4.11.1.1.5 Column Statistics

Statistics for each column are stored in the cache when the database is running, and in the main database file when the database closes down to preserve the values.

4.11.1.1.6 Table Page Map

The table page map stores all the pages that make up a given table.

4.11.2 Engine Task Stacks and Data Structures

Inside the ASA database engine, there is a multi-tasking kernel running. This kernel uses Windows NT fibers to execute request tasks. Stack memory is allocated by the OS when the fibers are created. A descriptor structure is used for each task.

4.11.3 Communications Buffer Pool

The Communications Buffer pool is a set of buffers used for communications. It is allocated from the system on startup. Each connection to the server uses two buffers. Active requests use at least one more. Each request is received from the client into this buffer pool, and the buffers remain allocated to that request until the request has finished processing in the engine. Replies to the clients are also stored in buffers from this buffer pool. As each response buffer is acknowledged by the client, it is released. The last response buffer is not released back into the pool until the next request is received.

4.12 Database Connections

The connection between the user's client application process and ASA is through NT's named pipes. Named pipes allow interprocess communication on a single machine or across networks. There are only two ends to a pipe: a server end and a client end. Each end is identified and accessed with a different handle. Only one client can connect to the client end of a pipe at a time. In general, the semantics of using named pipes is similar to using a file; Create/Open, Read, Write, and Close operations are performed by the client and the server with handles to the pipe.

ASA creates a named pipe with the CreateNamedPipe function call. The call specifies a name for the pipe and a maximum number of instances for the named pipe, and returns a handle to the server end of the pipe. (CreateNamedPipe can be called repeatedly with the same name to create any number of instances of the named pipe as long as the number does not exceed the specified maximum number of instances.) The name of the pipe is the server name. The server name is given using the "-n" switch, or defaults to the name of the first database on the command line when ASA is started. (If ASA starts without loading a database file, then it must be given a name using the "-n" switch.) The named pipe created by ASA is bi-directional, allows overlapped I/O, and is message oriented. After creating the named pipe, ASA calls ConnectNamedPipe to determine when a client has connected to the pipe.

The client application connects to the pipe with a CreateFile function call. The client application must know the name of the pipe. This function call returns a handle to the client end of the named pipe started by ASA. When the client connects and ASA's ConnectNamedPipe call returns, ASA makes a second instance of the pipe by calling the CreateNamedPipe function call with the same name, and then calling ConnectNamedPipe again on that new instance. Each call to CreateNamedPipe returns a different handle to the server end of the new instance of the pipe created by the call. Once a client opens the client end of a named pipe instance, that instance cannot be opened by another client. In this way, multiple, concurrent connections from clients can be serviced by ASA securely.

After the ConnectNamedPipe call signals the event object, the physical connection between the client process and the server is established. Communications between client and server continue to use the private named pipes for the duration of the connection.

When a client disconnects, the instance of the pipe used by the client is deleted. ASA calls DisconnectNamedPipe function, which invalidates the client handle. ASA then calls CloseHandle, which invalidates the server end handle to the pipe.

4.13 Kernel

4.13.1 Threads, Fibers, and Tasks

There are four threads running in ASA:

- Main engine,
- Named pipes server monitor, and
- Engine notifier.

The main engine thread processes requests. Each active request is assigned to a request task when it is received (as opposed to tasks being permanently assigned to a connection.) ASA schedules request tasks using Windows NT fibers. Fibers are subsets of threads that are scheduled by ASA rather than Windows NT. Each fiber has its own stack that is allocated by Windows NT. The number of request tasks governs the number of active requests that can be worked on simultaneously by the database engine. The number of request tasks does not restrict the number of users, or the number of requests that can be sent to the engine. The switch restricts the number of requests that

the engine works on simultaneously. If there are more outstanding requests, they are queued, and when an active request completes, the next request is picked up from the queue. The default number of request tasks for Windows NT is 20.

The named pipes server monitor waits for connection attempts.

The engine notifier checks the streams environment periodically to delete old connections, check for new connections, and do liveness checks.

4.13.2 Internal Data Structures

ASA implements its internal data structures as objects in C++.[4] Some of ASA's major internal data structures include the following:

- **Engine:** contains list of active databases, ID of default database, routines to find the maximum number of allowed connections, and to shut down the server,
- **Database:** contains status, connection list, values of public options, pointer to store, list of locks, list of files, list of users, and a list of tables,
- **Connection:** contains pointer to streams connection object, user, cursors, temp table list,
- **Task:** contains state, pointer to stack, fiber handle, and
- **Cache:** manages the cache.

In addition, ASA stores representations of SQL objects (such as tables) in memory, when it is executing. ASA implements its own mutex mechanism to synchronize access to its internal data structures among executing fibers.

4.14 Query Processing

This section describes the processing of a SQL statement after a request is received from the Command Sequence (CmdSeq) protocol. The steps are depicted in Figures 4.5 and 4.6, SQL Processing. The first step in the processing of a SQL statement is the creation of a parse tree. If the SQL statement is a Data Definition Language (DDL) statement (e.g., ALTER, CREATE, DROP, or GRANT), ASA immediately checks the permissions and executes the query. If the statement is a Data Manipulation Language (DML) statement (i.e., SELECT, INSERT, UPDATE, or DELETE), ASA annotates the parse tree with information about the referenced object. As part of annotating the parse tree, ASA checks that the referenced tables exist, finds their object IDs, and associated columns.

[4] These are internal objects, not the same as Named Objects protected by DAC. These objects are not accessible at the user interface.

During annotation of the parse tree, ASA performs the Discretionary Access Control (DAC) permission checks. The checks for *INSERT* and *DELETE* permission are just a check on a single table, since ASA allows INSERT and DELETE on just a single table at a time, even if they are performed through a view. An UDATE operation may be performed on rows in more than one table in a single statement. In addition, *UPDATE* permission can be granted on a subset of the columns of a table. If there is a WHERE clause in the UPDATE statement, the rows to be updated must be selected.

For SELECT statements, subqueries may be nested inside queries. For each SELECT node, there are pointers to table structures for each table referenced and a list of columns. Permissions are checked by comparing the requested operation and the objects referenced with the lists of permissions pointed to by the User structure. ASA uses the following algorithm for checking object permissions:

1. If the user is a DBA, grant access, else
2. If the user is the owner of the object, grant access, else
3. If the user ID has been granted the permission, grant access, else
4. If the user is a member of group that has been granted the permission, grant access, else
5. Deny access.

If the SQL statement passes the permission check, it passes through the pre-optimizer and the access plan is built. The access plan is a structure that is interpreted. The access plan is sent through the optimizer and then the query is executed. Note that the permission check is performed once for each SQL statement before it starts executing. For example, if a SELECT statement passes the permission check and starts retrieving rows, it will not stop retrieving rows if the user's SELECT permission is revoked from the table.

Cursors are used to execute SELECT statements from **dbisqlc** and through Embedded SQL (ESQL) in application programs. There are several SQL statements related to cursors. The sequence of cursor operations is DECLARE, OPEN, FETCH, and CLOSE. The SQL Statement is parsed and the permission checks are performed when the OPEN statement is executed. When the FETCH statement is executed, no additional access control checks are made. As shown in Figures 4.3 and 4.4 - SQL Processing, if the SQL statement is a cursor FETCH, control immediately branches to the Execute block.

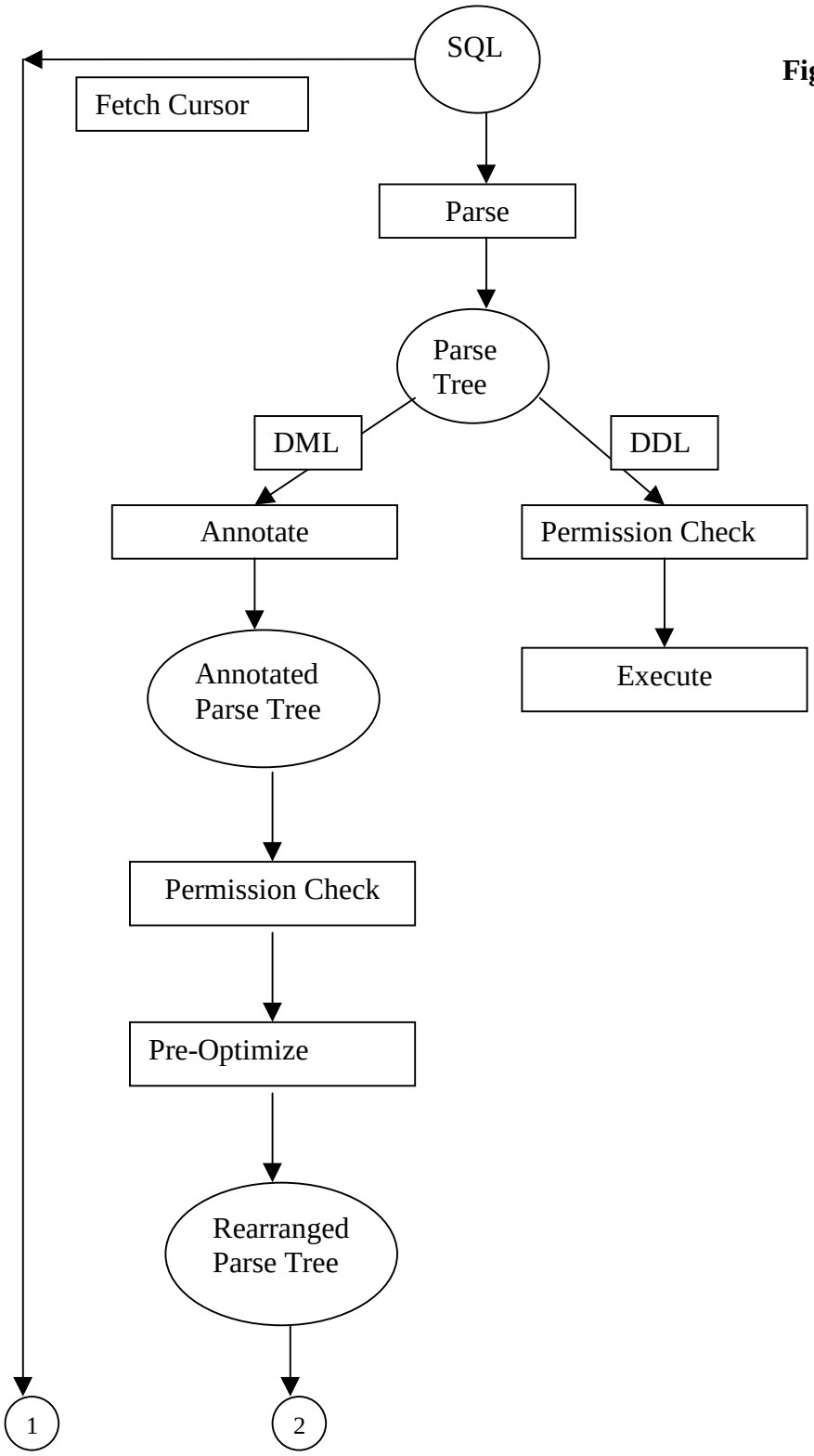
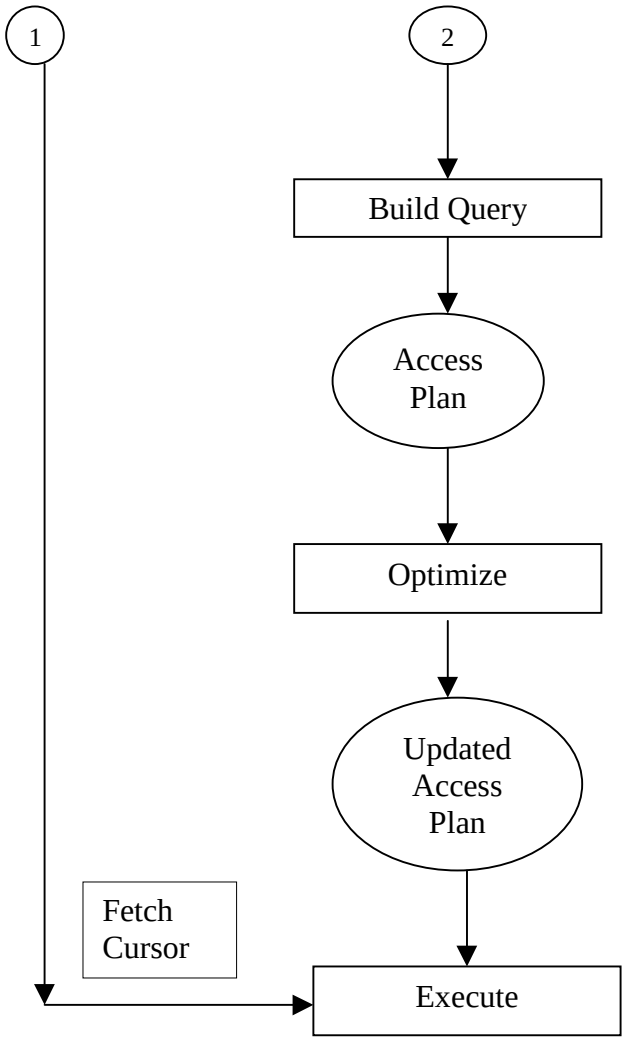


Figure 4.3 - SQL Processing

Figure 4.4 - SQL Processing (Concluded)



4.15 Transaction Processing

ASA supports transaction processing to ensure:

- **Transaction Atomicity**: SQL statements logically related as a transaction are executed as one unit of work.

- **Transaction Concurrency**: Concurrent transactions against the same data can be executed with consistent and correct results.

- **Transaction Recoverability**: Data can be properly recovered if system failure occurs before committed transactions are written to the database file, or if system failure interrupts the execution of one or more transactions.

4.15.1 Transaction Atomicity

In a relational DBMS, the tasks making up a transaction are defined in SQL statements. ASA executes all or none of the SQL statements in a transaction to guarantee the consistency of data in the database.

Transactions start with the first statement following a connection to a database, or the first statement following the end of a transaction.

Transactions complete with one of the following events:

- A COMMIT statement makes the changes to the database permanent.
- A ROLLBACK statement undoes all the changes made by the transaction.
- A statement with a side effect of an automatic commit is executed. Data Definition Language (DDL) statements such as ALTER, CREATE, COMMENT, and DROP all have the side effect of an automatic commit.
- A disconnection from a database performs an implicit rollback.

4.15.1.1 Commit and Rollback

A COMMIT statement makes the changes to data in the database permanent, when it is determined that all transaction tasks have been completed successfully.

A ROLLBACK statement undoes the changes to data in the database when it is determined that some transaction tasks cannot be successfully completed.

4.15.1.2 Savepoint

A savepoint delimits a group of logically related tasks within a transaction. A SAVEPOINT statement initiates a savepoint. A savepoint can be named by assigning an identifier in the SAVEPOINT statement. Savepoint names permit savepoints to be nested. Using named, nested savepoints creates multiple active savepoints within a transaction. All savepoints are released when

a transaction ends. Changes between a SAVEPOINT and a RELEASE SAVEPOINT can be canceled by rolling back to a previous savepoint or rolling back the transaction itself. Changes made anywhere in a transaction, including changes between a SAVEPOINT statement and a RELEASE SAVEPOINT statement, become a permanent part of the database only when the transaction itself is committed.

4.15.2 Transaction Concurrency

In multi-user databases, concurrent transactions may interfere with each other as they attempt to perform operations against the same row in the same table over the same period of time. This section first describes the types of possible inconsistencies within a transaction and then describes how locks and isolation levels can be used within ASA to prevent them.

4.15.2.1 Types of Data Inconsistencies within Transactions

The following three types of inconsistencies can occur when transactions share the same database data and read locking is disabled:

- Dirty read: Occurs when a transaction reads a row modified or inserted by a concurrent transaction that has not yet committed the transaction.

- Non-repeatable Read: Occurs when a transaction reads the same row more than once and obtains different results each time because a concurrent transaction modified the row in the interim and committed the change.

- Phantom row: Occurs when a transaction reads a set of rows satisfying a certain condition more than once and obtains different result sets each time, because in the interim a concurrent transaction inserted, deleted, or updated one or more rows meeting the same condition.

Note that data consistency here refers to data held by a transaction, not to data committed to the database.

4.15.2.2 Locking

To prevent concurrent transactions from interfering with each other, ASA implements automatic row-level locking. A lock is acquired by a transaction on a table row (or index) for either a write or a read operation. Locks have the effect of ensuring that the data that a transaction is using does not change in an unpredictable way. Locks are held until the transaction ends with a COMMIT or a ROLLBACK.

ASA uses three types of row-level locking:

- **Write lock:** A transaction automatically acquires a write lock on a row when it performs a write operation (INSERT, UPDATE, or DELETE) on it. A write lock is an exclusive lock. No other transaction can obtain any kind of lock on a row with a write lock. A transaction cannot obtain a write lock on a row with any kind of lock. Write locks ensure that write operations on given data are performed exclusively by one transaction until it is committed or rolled back.

- **Read lock:** A transaction may acquire a read lock on a row of a table when it performs a read operation (i.e., SELECT) on it. A read lock is a shared lock. Other transactions can acquire read locks on a row with a read lock.

- **Phantom lock:** A phantom lock is a read lock on a set of rows and the indexes used to obtain the set. A transaction acquires a phantom lock on every row in a set of rows produced by a read operation. The transaction also acquires a read lock on every index used for the read operation. A phantom lock is a shared lock.

Locks for write operations are mandatory. A transaction does not have to request a write lock. The lock is automatically requested when a write operation is executed. Locks for read operations are not mandatory. Requests for read locks can be controlled through isolation levels. At some isolation levels, no read lock is requested. At other levels a read lock is automatically requested when a read operation is executed.

Exclusive locks create a potential for conflict between concurrent transactions. Deadlock arises when two or more transactions contend for the same resource. It occurs only when the BLOCKING option is set to ON and transactions block. There are two types of deadlock: cyclical blocking and thread blocking. Cyclical blocking occurs when two or more transactions hold resources needed by each other and wait for each other to a release a needed resource. Thread blocking occurs when all the request tasks in the database engine are blocked. In both types of deadlock, ASA automatically detects the deadlock and rolls back the last blocked transaction, which resolves the deadlock. Note that the higher the isolation level, the greater the likelihood of blocking resulting in an impact on performance.

4.15.2.3 Isolation Levels

Isolation levels permit read operations on given data over a range of conditions and guarantees. ASA supports the four ANSI standard isolation levels. Descriptions of the ANSI isolation levels and their ODBC equivalents are shown in Table 4.13.

Table 4.13 - Isolation Levels

ANSI	ODBC	Description
0	RU	Read uncommitted
1	RC	Read committed
2	RR	Repeatable read
3	TS	Transactions serialized

Isolation levels provide for a trade-off between concurrency and consistency. Isolation level 0 offers the greatest degree of concurrency among transactions performing read and write operations and the least guarantee that a transaction is working with consistent data. Isolation level 3 offers no concurrency among transactions performing read and write operations—the transactions are serialized—and the greatest guarantee that a transaction is working with consistent data.

- **Isolation level 0** (RU) is the default. It allows dirty reads, non-repeatable reads, and phantom rows to occur in the application. No read locking is in effect with level 0. At Isolation Level 0, a

read operation is permitted on a row whether or not it has a write lock. There is no guarantee that, during the life of the transaction performing the read, the row will remain unmodified by a concurrent transaction; or that the row, if already modified by a concurrent transaction, will not be rolled back.

- **Isolation level 1** (RC) prevents dirty reads. It allows non-repeatable reads and phantom rows. At Isolation Level 1, a read operation is permitted on a row if it has no write lock. A read lock is acquired on the row. The lock lasts only for the *duration of the read operation on the current row*, not for the life of the transaction. If the transaction returns to the row for another read operation, there is no guarantee that the data will be the same. Since the read lock does not persist for the entire transaction, another transaction can acquire a write lock and modify it in the interim.

- **Isolation level 2** (RR) prevents dirty reads and non-repeatable reads. It allows phantom rows. At Isolation Level 2, a read operation is permitted on a row only if it has no write lock. A read lock is acquired for the row and *persists until the transaction ends* with a commit or rollback. The persistence of the read lock for the life of the transaction guarantees that every subsequent read by the transaction will return the same data.

- **Isolation level 3** (TS) prevents dirty reads, non-repeatable reads, and phantom rows. At Isolation Level 3, a read operation is permitted on a *set of rows* only if it has no write lock. A read lock is acquired for every row in the result set and for each index used to return the result set. All of these read locks persist for the duration of the transaction. This guarantees that every subsequent read by the transaction will return the same data in each row and the same rows in the result set.

4.15.3 Transaction Recoverability

ASA supports transaction recovery so that transactions committed before a system failure can be recovered and made permanent in the database. Transactions uncommitted before a system failure or in progress at the time of system failure are rolled back. When ASA restarts after system failure, it automatically recovers data to a consistent state using three logs:

- Checkpoint log
- Transaction log
- Rollback logs

ASA uses the checkpoint log to restore database pages to a known state at a point in time prior to system failure. The checkpoint log stores copies of database pages before they are updated. The checkpoint log is stored in a database file.

ASA uses the transaction log to apply transactions that were executed after the point in time of the last known state. The transaction log stores all changes to the database in the order that they occur. Inserts, updates, deletes, commits, rollbacks, and database schema changes are all logged. The transaction log is stored in the transaction log file.

ASA uses rollback logs to undo transactions that were incomplete or uncommitted when system failure occurred. ASA maintains a rollback log for each active transaction. When a SQL statement in the transaction modifies a row in a database table, a copy of the row before it is modified is saved in the rollback log. A rollback log exists for the life of the transaction, and it is deleted when the transaction is committed or rolled back. ASA caches rollback logs in memory and writes them to the main database file.

4.16 Java Virtual Machine

The ASA Java Virtual Machine (JVM) is a software application that provides a mechanism for stored procedures written in Java to access relational data stored in ASA databases.

The ASA JVM is an abstract computing machine that is compliant with Sun Microsystems' JVM specification. Like a real computing machine, it has an instruction set. JVM instructions are called bytecodes. The Java programming language is compiled to these bytecodes. Java is an object-oriented, interpreted programming language based on C/C++ that was developed by Sun Microsystems. The ASA JVM runs Java programs by interpreting these bytecodes and translating them to machine instructions that will execute on the underlying operating system and/or hardware platform. The ASA JVM supports a subset of the API defined by Sun's Java Development Kit (JDK) version 1.1.6. The ASA Java interface also includes APIs defined by Sybase and is described in Section 4.7.3, Java Application Programming Interfaces.

4.16.1 ASA JVM Implementation

The ASA JVM is implemented as a Dynamic Link Library (DLL) and is executed as part of the ASA process. DLLs are files that contain one or more functions that are compiled, linked, and stored separately from the processes that use them. The operating system maps the DLLs into the address space of the calling process when the process is starting or while it is running. The operating system controls access to the ASA JVM DLL entry points as well as the file.

4.16.2 ASA JVM Architecture

Java-enabled ASA databases use multiple ASA JVM instances and separate heaps within a single address space to maintain separate domains for each user. Underlying ASA memory management mechanisms are relied upon to provide separation between each heap.

4.16.3 ASA JVM Self-Protection

ASA JVMs protect themselves and offer additional protections from the bytecode they are interpreting by implementing what is called a sandbox security model. The sandbox model limits certain actions that bytecode can perform (e.g., reading and writing to the local disk).

ASA JVM sandboxes are maintained by a collection of components within the virtual machine, including the mechanism that loads classes, the mechanism that verifies class file formats, runtime mechanisms that control access to memory, and mechanisms that control access to outside of the ASA JVM.

The ASA JVM mechanism that loads Java classes into memory is called the class loader. The class loader prevents naming conflicts between classes loaded into the Namespace Heap and prevents these classes from being replaced by Java applications running on any given Connection VM. Both naming conflicts and class replacement are prevented by maintaining a list within each class loader object called a namespace. The namespace consists of a set of unique names for classes that have been loaded by a given class loader. New class loader objects are instantiated (1) when the Peer VM is instantiated and (2) after new classes have been installed by a DBA, a new connection to the database has been established, and a reference to a Java class definition by that new connection has been made. After new class loader objects are instantiated, the existing class loader objects are deleted after the last connection that is referring to any of its class definitions is dropped.

The ASA JVM mechanism that verifies class file formats is called the class file verifier. After a class loader object loads any given class, the class file verifier checks the integrity of that class's bytecodes and checks jump instructions against symbolically referenced classes, fields, and methods. The class file verifier also performs runtime checks to verify class, method, and variable visibility (i.e., enforces public/private/protected visibility modifiers).

The ASA JVM runtime mechanisms that controls access to memory include type-safe reference casting, no pointer arithmetic, no ability to explicitly free allocated memory, array bounds checking and checking of references for null. There are no pointers in Java, and JVM garbage-collection mechanisms are relied upon to free allocated memory (when there are no more references to an object, a low-priority thread running in the background frees the object's allocated memory).

The ASA JVM mechanisms that control access to outside of the ASA JVM (i.e., to the underlying DBMS and/or operating system) consists of the set of Java native methods. Java native methods are non-Java applications and libraries that Java applications have the ability to invoke using the ASA JVM's Java Native Interface (JNI) mechanism. Java native methods are relied upon to provide access to the rest of the TCB. It is important to note that the ASA Java implementation does not use a security manager, as defined by the JDK 1.0 specification. Instead, the ASA DBMS is relied upon to control access to outside of the ASA JVM (the connection structure of the worker thread is used by the ASA DBMS to make access control decisions). None of the java.lang.SecurityManager class' native methods are implemented. They throw exceptions when invoked. (The SecurityManager is not used by ASA to implement C2 requirements.) The SecurityManager is included for backward compatibility.

4.16.4 ASA JVM TCB Interface

The ASA JVM presents a TCB interface by providing a mechanism for stored procedures written in Java to access relational data stored in ASA databases.

Java classes are stored in ASA databases as data types that can be used as the data type of a column in a table or a variable. However, while runtime Java classes (runtime classes are defined as the set of classes included in the 'java', 'sun', 'com.sybase', and 'sybase.sql.asa' packages) can be used in the same way, they are not stored in the database. Runtime Java classes are stored in the 'Java' subdirectory of the ASA installation. Java native method implementations (i.e. non-Java applications and libraries) are not stored in the database either.

Java class instantiations (objects) can be saved as a value in an ASA table cell. Java objects can be inserted into a table, SQL SELECT statements can be executed against the fields and methods of the Java objects stored in a table, and Java objects can be retrieved from the table.

The ASA JVM interprets Java bytecode. Java bytecode instructions can be divided into several categories:

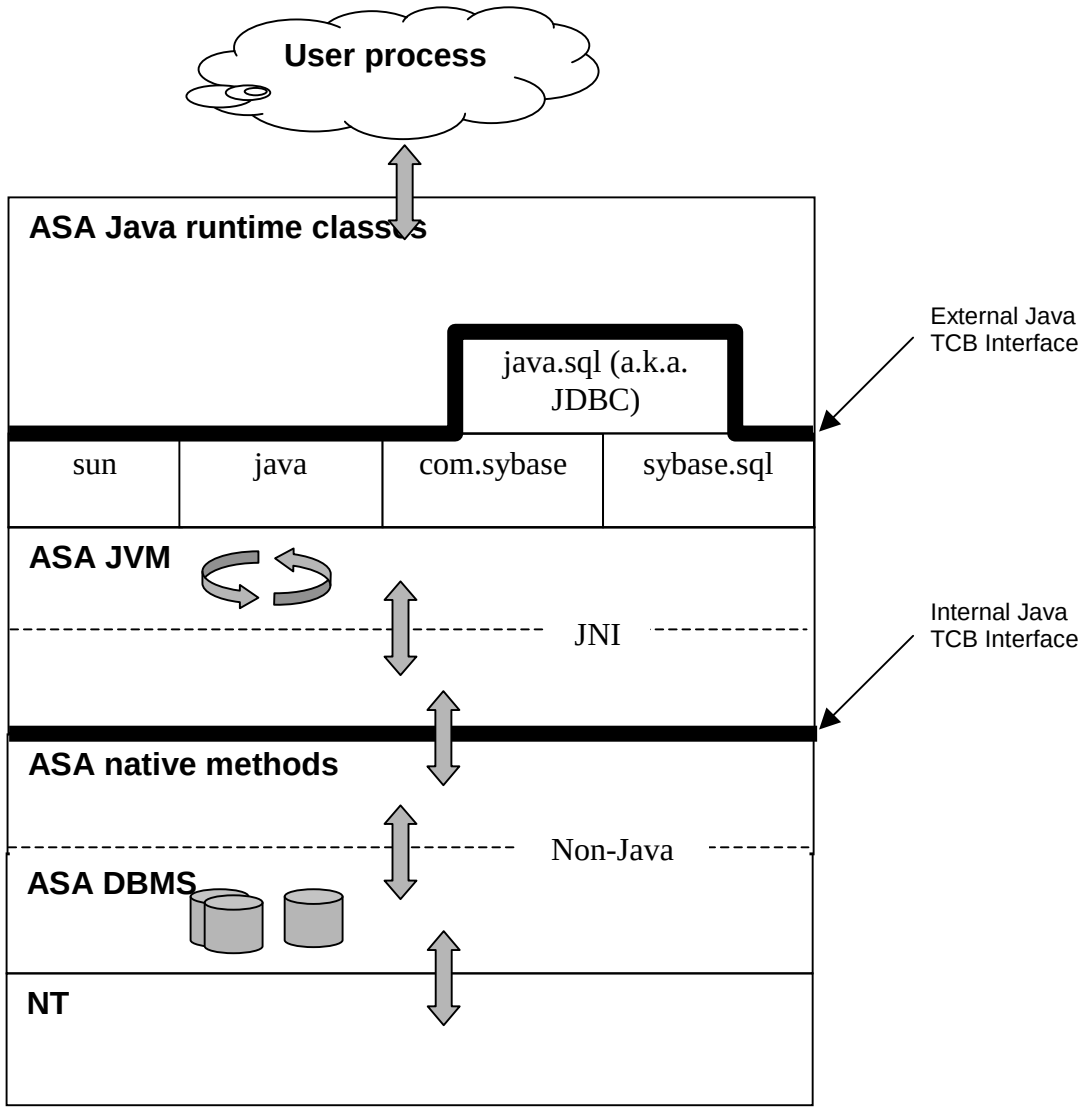
- Pushing constants onto the stack
- Accessing and modifying the value of a register
- Accessing arrays
- Stack manipulation (e.g., pop the top word from the operand stack, swap the top operand stack two words)
- Arithmetic, logical, and conversion instructions
- Control transfer
- Function return
- Manipulating object fields
- Method invocation
- Object creation
- Type casting

When the Java bytecode being interpreted makes a call to interact with ASA databases using the ASA JVM native method declaration mechanism, the ASA JVM native method declaration mechanism allows the invocation of Java native methods (non-Java applications and/or libraries) to pass requests to and receive responses from the ASA DBMS.

It follows, then, that the TCB interface that the ASA JVM presents consists of the runtime classes' compiled bytecodes. The ASA JVM itself supports the TCB interface by providing a mechanism to interpret bytecodes and to provide a runtime environment (Java is an interpreted language).

Given there is a one-to-one correspondence between class definitions and corresponding, compiled bytecodes, the APIs defined by the set of ASA Java runtime classes comprise the external ASA JVM TCB interface. Similarly, the ASA Java native method implementations comprise the internal ASA JVM TCB interface by providing the interface between ASA JVMs and the rest of the TCB. Figure 4.5 depicts these interfaces graphically.

Figure 4.5 - ASA logical Java interfaces



Underlying ASA DBMS mechanisms are relied upon to protect database data and underlying operating system mechanisms are relied upon to protect applications and data under the control of the operating system from the Java native methods. Java native methods do not make any direct calls to the underlying operating system. The DBMS is relied upon by the Java native methods to provide services such as memory management that would otherwise be provided by the underlying operating system. Specifically, the DBMS is relied upon to define and implement services that are declared in what is called the ASA JVM host function table. The ASA JVM host function table is a C++ structure that defines the functional interface that ASA JVMs use to request services such as memory management, threads, semaphores, reading class files, and sockets.

Requests made by Java bytecode to access ASA database data are executed with the permissions of the connection issuing them. Also, because Java classes are installed in ASA as data types, there is no equivalent of the GRANT EXECUTE statement. There is no concept of ownership with respect to Java classes.

It is important to note that the networking API (i.e., the 'java.net' package) does not present an additional set of TCB interfaces. An additional set of TCB interfaces is not presented because all Java socket-related functionality was disabled in the evaluated configuration by updating the ASA JVM host function table. Datagram, multicast, and stream sockets are not supported in the evaluated configuration. The 'java.net' package cannot provide any networking functionality without this support. The 'java.net' package is included in the evaluated configuration for ease of packaging. Modifying socket-related parameters in the ASA JVM host function table effectively disables both the internal and external socket-related Java TCB interfaces that would otherwise be included in the evaluated configuration.

4.16.5 ASA JVM TCB Interface Configuration

New ASA databases are Java-enabled by default:[5] the ASA JVM is installed along with all of the Java runtime classes when a new ASA database is created; system tables are also updated with a list of available classes. There is also a mechanism that can be used by DBAs to install additional classes into ASA databases.

It is important to note that the installation of additional, non-runtime Java classes (i.e., those classes that are not included in the 'java', 'sun', 'com.sybase', and 'sybase.sql.asa' packages) into ASA databases does not invalidate the evaluated configuration. Native methods make the actual requests to and receive the actual responses from the underlying DBMS (and operating system, through the DBMS). In other words, native methods are the entities in the ASA Java implementation that are performing the actual security-relevant events. The installation of non-runtime classes does not invalidate the evaluated configuration. This is because ASA only supports the installation of native methods that are in the set of those relied upon by the set of runtime classes installed as part of the ASA software. Also, Java classes can only initiate requests to and receive responses from the underlying DBMS (and operating system, through the DBMS),

[5] Java can be disabled in databases when they are created by specifying the appropriate options/switches using either the CREATE DATABASE SQL statement or the dbinit.exe command-line database initialization utility.

Classes can be installed one at a time or as a set contained in a Java ARchive (JAR) file. DBA authority is required to install classes into ASA databases using Interactive SQL to issue the INSTALL statement. The steps involved in creating and installing non-runtime Java classes are as follows:

- Define the Class or set of Classes. Write Java Code that defines the class or set of Classes.
- Name and Save the Class or set of Classes. Save the code in files with an extension '.java'.
- Compile the Class or set of Classes. The compiler performs compile time checking and converts source code into class files that contain bytecodes.
- [OPTIONAL] Create a JAR file. JAR files can be created to hold one or more class files.
- Install the Class or set of Classes. Only DBA can install classes into the database.
- Use the Class or set of Classes. User can access the class to create Java objects.

Newly installed Java class definitions are only accessible by connections that are established after the class or set of classes has been installed. For redefined classes, existing connections refer back to the class definitions already in memory. For new classes, a counter mechanism is used to preclude access to the classes installed after the establishment of the connection. The counter mechanism works by numbering individual classes as they are installed. When a connection is created, it remembers the number of the most recently loaded class. When the Connection JVM asks the Peer JVM to load a class, if the class number is greater than the number saved in the connection structure, it will act as if the class is not found. This is because after a class definition has been loaded by an instance of the ASA JVM, it is held in memory until the associated connection is closed.

5 TCB Protected Resources

This section describes the users, subjects, and objects controlled by the Server.

5.1 Users

The security relevant attributes of a User are as follows:

- User ID,
- User Name,
- Password,
- Authorities,
- Group Permission,
- Group Membership,
- Table Permissions, and
- Procedure Permissions.

The User ID is an integer and the User Name is a string containing a unique name for the User ID. The DBA uses the User Name to create a User and ASA automatically assigns a unique User ID as a primary key. Although users are required to use integrated login and not passwords in the evaluated configuration, a user must still be created with a password. It is possible to create a User with a NULL password, and this prevents anyone from connecting as that User. (For more information on passwords, see Section 6.1, Identification and Authentication.)

Table permissions and procedure permissions are object permissions that a user has been granted on database objects. Object permissions are discussed in Section 5.3, Objects.

5.1.1 Authorities

The ASA supports two authorities in the evaluated configuration: DBA and Resource authority. DBA authority is only granted to trusted administrators and allows the user to manage the security functionality of the database. When a database is first created, the only user name that can login is "DBA". The TFM instructs the DBA to set the minimum password length to six and to change the password. The DBA can create additional individual DBA accounts by creating new users and granting them DBA authority. Users with DBA authority are the only trusted users in ASA. For convenience, a user with DBA authority is often referred to as the DBA, although there may be more than one user with DBA authority. Resource authority allows a user to create database objects such as tables, views, stored procedures, and triggers.

There are two other authorities, Remote DBA and Publisher, that are used to support database replication (which is not part of the evaluated configuration). The TFM warns the DBA not to grant Remote DBA authority in the evaluated configuration. The Publisher authority serves no purpose in the evaluated configuration, but granting it to users without the Remote DBA authority does not create a security vulnerability.

5.1.2 Groups

ASA provides a group mechanism to facilitate the granting and management of object permissions. A group is implemented as a User ID that has been granted the *GROUP* Permission. A Group can have members. A User ID that has been granted *GROUP* permission is also known as a Group ID. Membership in a group can be granted by either the user with the Group ID or a user with DBA authority. Users may be granted membership in more than one group, so the user-to-group membership is many-to-many. A group can also be a member of a group. A hierarchy of groups may be constructed, each inheriting permissions from its parent group.

Permissions are granted to a group in the same way that permissions are granted to individual Users. When object permissions are granted to a group, all members of the group inherit these permissions. However, DBA authority, Resource authority and Group permission are not inherited. They must be granted individually to each individual User requiring them. Also, ownership of database objects is associated with a single User ID and is not inherited by group members. If a Group ID creates a table, the Group ID is able to make changes to the table and grant object permissions on the table to other Users. Other User IDs who are members of the group are not owners of the table and do not have these rights. A Group ID (or DBA) must explicitly grant itself object permissions on objects that it owns in order to grant access to these objects to members of its group.

There are three special groups in ASA: SYS, PUBLIC, and DBO. All three of these groups have been created with a NULL password, so that no one can connect to them. The SYS group is the owner of the system tables and views for the database. All users are created as members of the group PUBLIC. The PUBLIC group is a member of the SYS group. DBO is the owner of many system stored procedures and catalogs.

5.1.3 Creating and Deleting Users

Users are creating using the GRANT CONNECT SQL statement. Then an NT account name must be mapped to the User Name using the GRANT INTEGRATED LOGIN SQL statement. For more information on integrated login, see Section 6.1, Identification and Authentication. Only the DBA can create users in ASA. The GRANT command is used for several purposes in ASA as follows:

- To create users,
- To define an integrated login,
- To grant DBA and Resource authority,
- To grant group permission,
- To grant group membership, and
- To grant object permissions.

The REVOKE statement is used to drop User IDs and revoke authorities and permissions. More information about granting and revoking object permissions is provided in Section 6.2, Discretionary Access Control.

5.1.4 Data Structures

Information about users is stored in the SYSUSERPERM system table, as shown in Table 5.1

Table 5.1 - SYSUSERPERM System Table

Column Name	Column Type	Description/Comment
user_id	smallint	Primary key
user_name	char (128)	
password	binary (36)	Stored encrypted
resourceauth	char(1)	Y/N - Has Resource Authority
dbauth	char(1)	Y/N - Has DBA Authority
scheduleauth	char(1)	Not used in evaluated configuration.
publishauth	char(1)	Not used in evaluated configuration
remoteddbaauth	char(1)	Not used in evaluated configuration
user_group	char(1)	Y/N - Has group permission
remarks	longvar	

Two other system tables store information about users:

- SYSGROUP - stores group memberships, and
- SYSLOGIN - stores information for integrated logins.

After a user connects to a database, information is stored about the user in the user structure. The user structure is a linked list of User Definition Structures. The User Definition Structure contains the following fields: user ID, user name, authority flags, a pointer to a list of groups, a pointer to a list of table permissions, and a pointer to a list of procedure permissions.

5.2 Subjects

The subject in ASA is a connection object. A connection object is created at the time that a user logs in. A subject is deleted when a user process disconnects from the database or the engine closes a connection. The engine closes connections when instructed by a DBA or when shutting down. When the database receives a request over the connection, it assigns a task from the task pool to execute the request. If no task is available, the request is queued. Tasks are not permanently assigned to connections.

The connection object is represented inside ASA by the connection structure. The connection structure contains a pointer to the associated user definition that contains the security attributes of the user. The connection structure also contains links to pointers to connection structures for the session, presentation, and transport layers.

The security relevant attributes of a subject (connection) are:

- User ID,

- User Name,
- Authentication Flag,
- Authorities,
- Group Permission,
- Group Membership,
- Table Permissions, and
- Procedure Permissions.

See Section 6.2, Discretionary Access Control for more information about permissions.

5.3 Objects

5.3.1 Named Objects

Named objects contain information and can be shared among database users. The named objects in ASA are:

- Tables,
- Views,
- Procedures, and
- Functions.

Named objects have an owner. An object may be either owned by its creator or created and assigned ownership by a DBA. Discretionary Access Control is applied to all named objects, and that policy is described in Section 6.2, Discretionary Access Control.

5.3.1.1 Tables

Tables are made up of rows containing data. Tables are created with the CREATE TABLE statement and are deleted by the DROP TABLE statement. The permissions that can be granted on a table are: *SELECT, INSERT, UPDATE, DELETE, ALTER* and *REFERENCES*. Columns are considered part of the table named object. *REFERENCES, SELECT,* and *UPDATE* can be granted for specific columns of a table. Table definitions are stored in the SYSTABLE system table, and column definitions are stored in the SYSCOLUMN system table.

An index may be created on a column or columns of a table to improve database performance. Indexes are created with a CREATE INDEX statement and deleted with a DROP INDEX statement. In order to create an index, a user must be the owner of the table, have *REFERENCES* permission, or have DBA authority. Indexes are considered an attribute of the table named object. Index definitions are stored in the SYSINDEX system table.

A primary key is a column, or set of columns, whose values uniquely identify every row in the table. A foreign key references a primary key in another table and is used to implement an integrity constraint. Foreign keys are created with the CREATE TABLE or ALTER TABLE statement. The ALTER TABLE statement may be used to delete a foreign key. A user may issue an ALTER

TABLE statement if the user is the table owner, if the user has *ALTER* permission on the table, or if the user has DBA authority. Information about foreign keys is stored in the SYSFOREIGNKEY and SYSFKCOL system tables.

A trigger is a set of SQL statements associated with a table that executes automatically when a specific INSERT, UPDATE, or DELETE statement is executed on the table. Triggers may be used to enforce a referential integrity policy on a database. In addition, triggers allow changes to cascade through related tables, to aid in enforcing complex column restrictions, or to perform actions in response to data modifications. Triggers are created with the CREATE TRIGGER statement. In order to create a trigger, a user must have resource authority and either be the owner of the table or have *ALTER* permission on the table. Triggers are considered table attributes, and they are protected by the access controls on the table. Trigger definitions are stored in the SYSTRIGGER system table.

5.3.1.2 Views

Views are derived tables consisting of rows and columns from base tables and other views. Views are defined using a SELECT statement. A view may select a subset of columns or rows, or join data from multiple tables. Views may be used to give users a customized perspective of the data and/or to enforce a security policy by limiting access to a subset of a table. Views are created with the CREATE VIEW statement and deleted with the DROP VIEW statement. Permissions that can be granted on views are: *SELECT, INSERT, UPDATE*, and *DELETE.* View definitions are stored in the SYSTABLE system table.

5.3.1.3 Procedures

Procedures are a series of Structured Query Language (SQL) statements to be executed as a unit. Procedures may be used to group a set of SQL statements for the convenience of users or to provide a controlled means of granting access to tables. Procedures are created with the CREATE PROCEDURE statement and deleted with the DROP PROCEDURE statement. Procedures are called using a CALL statement. *EXECUTE* permission can be granted on a procedure. Procedure definitions and parameters are stored in the SYSPROCEDURE and SYSPROCPARM system tables.

5.3.1.4 Functions

A Function is a procedure that returns a value. User-defined functions are created with the CREATE FUNCTION statement and deleted with the DROP FUNCTION statement. ASA User-defined functions are called in the same way as SQL functions using a SELECT statement. *EXECUTE* permission can be granted on a user-defined function. User-defined function definitions and parameters are also stored in the SYSPROCEDURE and the SYSPROCPARM system tables.

5.3.2 Object Study

The TCB interfaces, variables, and system tables variables were reviewed for potential objects. The disposition of the candidate objects is summarized alphabetically in Table 5.2. The candidate objects have been categorized as follows:

- Named objects as discussed above.

- Attributes of named objects, as discussed above.

- Public objects that can be read by any user, but can be written only by a user with DBA authority or by the system.

- TCB internal objects that are only accessible to a user with DBA authority.

- Objects that are local to a subject and cannot be shared.

- Not a data container. Although these may appear to be potential objects from the SQL syntax, they do not contain data.

- Not in the subset of protected objects. These are unprotected objects that can be written by one untrusted user and read by another. These objects have been reviewed and the lack of protection has been deemed acceptable at C2. They contain control information, not user data.

- Objects that are not part of the evaluated configuration such as objects related to replication.

Table 5.2 - Disposition of Candidate Objects

Candidate Object	Disposition
Article	Used for data replication. Not in evaluated configuration.
Batch	Not a data container.
Column	Part of a table
Comment	Comments are attributes of named objects or TCB internal objects.
Cursor	Local to a subject
Database	Can only be modified by a DBA. TCB internal object
Dbspace	Part of a database. TCB internal object
Descriptor	Allocates space within an application program to store data. Local to a subject
Domain	Attribute of user data type.
File	Some administrative functions result in the creation or deletion of operating system files. These functions can only be performed by a DBA. TCB internal object
Foreign Key	Table attribute
Function	Named object
Group	TCB internal object
Index	Attribute of a table
Java class definitions	Stored as table column data types and only a DBA can install. Neither a data container nor a TCB internal object.
Java methods (instance methods)	Stored functions that are local to a subject.
Java methods (class methods)	Stored functions that are global. Not a data container.
Java objects	Local to a subject.
Java objects (serialized)	Global: public object Local and connection-level: local to a subject.

Java socket objects, as defined in 'java.net'	Not supported in the evaluated configuration.[6]
Local Temporary Table	Local to a subject
Login	TCB internal object.
Message	Not in subset of protected objects.
Optimizer Statistics	Only DBA can delete statistics. TCB internal object.
Option	Database-wide options: public objects; User options: not in subset of protected objects.
Procedure	Named Object
Property	Public objects.
Publication	Used for data replication. Not in evaluated configuration.
Remote Type	Used for data replication. Not in evaluated configuration.
Remote User	Used for data replication. Not in evaluated configuration.
Schema	Not a data container
Subscription	Used for data replication. Not in evaluated configuration.
Table	Named Object
Transaction	Not a data container
Trigger	Attribute of a Table
User	TCB Internal Object
User Data Type	Not in subset of protected objects
Variables	Global: public object Local and connection-level: local to a subject (cannot be shared);
View	Named object
Write File	TCB Internal Object

The sections below provide more background information on the candidate objects and the rationale for their exclusion as named objects.

5.3.2.1 Databases

Databases contain tables that store data. Databases are made up of a collection of files that may include the database file, dbspace files, a transaction log, a transaction log mirror, and a write file. Databases can be created using the CREATE DATABASE SQL statement or the Initialization utility. Databases can be deleted using the DROP DATABASE SQL statement or the Erase utility. When a server is started, the -gu command line option can be set to control who has permission to create a database. The TFM instructs the administrator to start the server as an NT service. The default is that only a user with DBA authority has permission to create a database. Starting the server as a service ensures that an untrusted user cannot start the server and change the default. There are several SQL statements and database utilities that operate on a database and its associated files. A user must have DBA authority to perform these operations. Databases and their associated files have been categorized as TCB Internal Objects.

[6] If this were not the case, Java sockets would be named objects. Table data (including serialized Java objects) could then be passed between untrusted users in a way that would circumvent access control and audit policies.

5.3.2.2 Properties

Properties are system information that is gathered as the server is running and that is primarily used for performance tuning. Examples of properties are cache hits, disk reads, and packet counts. The information is not security relevant. There are three levels of properties:

- Server level properties that apply across the server as a whole.
- Database-level properties that apply to each database on the server.
- Connection-level properties that apply to each connection.

The following SQL system functions retrieve properties and return the requested value for the current connection:

- CONNECTION_PROPERTY,
- DB_PROPERTY, and
- PROPERTY for server properties.

The following system stored procedures also retrieve properties:

- sa_conn_info,
- sa_conn_properties,
- sa_db_info,
- sa_db_properities, and
- sa_eng_properties.

Properties values are read-only. Property values are set by the system and readable by all users. Properties are public objects.

5.3.2.3 Options

Options are used to set configuration parameters such as the DATE_FORMAT or the WAIT_FOR_COMMIT option. The scope of an option can be either database-wide or user-specific. Most options can be set for either the PUBLIC user ID or a specific user ID. If a database option is not set for an individual user, the database option for the PUBLIC User ID is used. Options can be either permanent or temporary. Permanent options persist until they are reset. Temporary options only last for the duration of the current connection.

Some options can only be set for the PUBLIC user ID. These are database-wide or public options. Three security-relevant public options are:

- LOGIN_MODE: **integrated** required in the evaluated configuration.
- MIN_PASSWORD_LENGTH: **6** or greater required in the evaluated configuration.
- AUDITING: **on** required to enable auditing. Note that the ASA audit mechanism meets the C2 audit requirement that the TCB be able to audit security relevant events. The audit mechanism can be disabled or enabled by the DBA.

Options are set with the SET OPTION statement. Users can set database options for their own connections. Only a user with the DBA authority can set the options for the Public User ID or another User ID.

Options are retrieved with the GET OPTION statement. There are no permission checks on the GET OPTION statement. Any user can read the database-wide options or the user-specific options for any user. Options can also be retrieved using the connection_property system function, as described in Section 5.3.2.2, Properties. Options are stored in the SYSOPTIONS system table, which is publicly readable.

Database-wide options are public objects, because they are set by the DBA and are readable by all users. An untrusted user can set a user-specific option for himself, that all users can read. Therefore, user-specific options are not in the subset of protected objects. They contain control information, not user data.

5.3.2.4 Variables

There are three types of SQL variables:

- Local variables,
- Connection-level variables, and
- Global variables.

Local variables are local to a statement. Connection level variables are local to a connection. Multiple statements can be executed within a single connection, so both are local to a single connection.

All variables are read-only. Local variables and connection-level variables cannot be shared. Global variables are set by the system and are public objects.

5.3.2.5 Cursors and Descriptors

Cursors provide access to a set of rows returned from a query. A cursor consists of two parts: a cursor result set and the cursor position. A cursor has a name and may be used to update or modify rows. Cursors are local to a connection and cannot be shared.

Descriptors are used to allocate data storage in an application program. They are local to a connection and cannot be shared.

5.3.2.6 Comments

Comments are attributes of the following objects: column, foreign key, index, login, procedure, publication, subscription, table, user, trigger, and view. Tables, views, and procedures are named objects. Columns, foreign keys, indexes, and triggers are part of a table object. Login and user are TCB internal objects. Subscription and publication are outside the evaluated configuration, because replication is not included in the evaluated configuration.

5.3.2.7 User Data Types

User data types can be created by the CREATE DOMAIN statement and stored in the SYSUSERTYPE system table. Any user with resource authority can execute the CREATE DOMAIN statement and all users have access to the data type. User data types are not in the subset of protected objects. They are used to store control information, not user data.

5.3.2.8 Messages

There are three message interfaces. The MESSAGE statement displays a message on the server message window. The CREATE MESSAGE statement creates a message to be used in print statements and error messages. The DROP MESSAGE statement deletes the message. Only the owner of the message or a user with DBA authority can drop a message. The text of the message is stored in the SYSMESSAGES system table and is publicly readable. Messages are not in the set of protected objects. Messages are used to store control information, not user data.[7]

5.3.2.9 Java Candidate Objects

Java classes are stored in ASA databases as data types that can be used as the data type of a column in a table or a variable. Java class instantiations (objects) can be saved as a value in an ASA table cell. Java objects can be inserted into a table, SQL SELECT statements can be executed against the fields and methods of the Java objects stored in a table, and Java objects can be retrieved from the table.

None of the Java candidate objects listed above can be categorized as named objects because they are either not data containers or they cannot be shared between subjects, the latter due to the exclusion of Java networking functionality in the evaluated configuration.

5.3.2.10 Replication Objects

SQL Remote and replication have not been included in the evaluated configuration. Therefore, the following candidate objects are not part of the evaluated configuration:

- Article,
- Publication,
- Remote type,
- Remote user, and
- Subscription.

[7] Note that messages were added to ASA to be compatible with ASE, the product evaluated as the Sybase SQL Server. Messages were deemed to be storage objects protected by labels at Class B1, but not named objects for ASE and this was acceptable to the TRB.

6 TCB Exported Security Policies

6.1 Identification and Authentication

ASA has two identification and authentication (I&A) mechanisms: integrated login and standard (password) login. Users are required to use integrated login in the evaluated configuration. The standard login mechanism provides protection for users with DBA authority, but is otherwise not used in the evaluated configuration.

6.1.1 System Tables

ASA uses two tables to implement I&A: SYSUSERPERM and SYSLOGIN. The SYSUSERPERM table has one row for each user. Note that the password column is used though integrated login is required in the evaluated configuration. This table also contains one column for each authority. Only the DBA and Resource authorities and *GROUP* permission are used in the evaluated configuration. See Table 5.1 for a depiction of the SYSUSERPERM table.

For integrated login to be used, the SYSLOGIN system table (Table 6.1) is also required. Integrated login relies on mapping NT accounts to ASA user names. It contains only two security-relevant columns, integrated_login_id for the NT username that serves as the table's primary key and login_uid, a foreign key that references the column, user_id, in the SYSUSERPERM table.

Table 6.1 - SYSLOGIN System Table

Column Name	Column Type	Column Description/Constraints
integrated_login_id	CHAR(128)	NT Account. Primary key, NOT NULL.
login_uid	UNSIGNED INT	ASA User ID. Foreign key references SYSUSERPERM user_id, NOT NULL.
remarks	LONG, VARCHAR	Comment string.

Like all system tables, SYSUSERPERM and SYSLOGIN are owned by user SYS and cannot be changed through the data manipulation statements (INSERT, UPDATE, and DELETE). In addition, users without DBA authority are not allowed to SELECT from the SYSUSERPERM and SYSLOGIN system tables.

6.1.2 Login Mode

ASA has three login modes: integrated, mixed, and standard. Only integrated login mode is allowed in the evaluated configuration. The TFM directs the administrator to set the Login Mode option to "integrated" during ASA installation.

6.1.3 Administration of User Accounts

For ASA, the responsibility of managing user accounts belongs to the DBA. Only the DBA can create a new database user with the "GRANT CONNECT TO user_name IDENTIFIED BY

password" SQL statement. The DBA can change an existing user's password with the same statement; every user also has the right to change their own password with this statement. Neither the DBA nor the non-DBA user has to verify their knowledge of the old password before changing the password with this statement. Only the DBA can delete a database user with the "REVOKE CONNECT FROM user_name" SQL statement.

6.1.4 Integrated Login

A user or client application connects to a database using integrated login by either setting the INTEGRATED parameter in the list of connection parameters to yes or by specifying neither a User ID nor a password in the connection string or connection dialog. If INTEGRATED=yes is specified in the connection string, an integrated login is attempted. If an attempt to connect to a database is made without providing a user ID or password, integrated login is attempted. The attempt succeeds or fails depending on whether the current NT account matches an integrated login mapping in the SYSLOGIN table. The current NT account is determined using calls to the NT Security Support Provider Interface (SSPI) as described in Section 3.8, Integrated Login.

6.1.5 Standard Login

Standard Login means supplying a username and password in order to connect. Users with DBA authority can always connect using their username and password. The TFM directs DBAs not to do this in the evaluated configuration, because integrated login is needed in order to be able to correlate the NT and ASA audit trails. Since the system does not prevent a user from connecting to a User ID with DBA authority using the password, the standard login mechanism is part of the evaluated configuration and users with DBA authority must protect their passwords.

To establish a connection with the server using standard login, the user must have already logged into NT. The user must then log into ASA to initiate an ASA session using a client application such as Interactive SQL (**dbisqlc**) by supplying his User ID and password. Upon receipt of the User ID and password, the db_connect function verifies that the User ID corresponds to an entry in the SYSUSERPERM table. If the entry is found, the password provided in the login request is hashed and compared with the hash of the User's password stored in the SYSUSERPERM table.

6.1.6 Password Management

The TFM requires that ASA be configured to enforce a minimum password length of six characters in the evaluated configuration. The cardinality of the minimum password space is 256^6, which is over 256 trillion possible passwords. There is no maximum length for passwords. Passwords are never stored; only the 36-byte hash of the password is stored. There is roughly a one in 2^{228} chance of two passwords hashing to the same value.

The password-hashing algorithm creates a binary 36-byte string from a random key, the User ID, and the password, and saves this value in the SYSUSERPERM table. (This value will be referred to as the password hash.) When encrypting a new password, a random 32-bit key is chosen. (This is done per password, not per database.) The hashing algorithm is such that the original key can be extracted from the password hash, but the User ID and password cannot be extracted. When authenticating a user, the key is extracted from the password hash, combined with the User ID and password, and then hashed into a 36-byte string. This 36-byte string is compared with the password

hash. If they match, the user is successfully authenticated. The random number is included in the hash to reduce the risk from dictionary attacks. Users cannot compare password hashes in different databases to guess matching passwords.

6.2 Discretionary Access Control Policy

The DAC mechanism allows the user to control access to named objects. The DAC mechanism includes GRANT and REVOKE statements and rules about group permissions. Object permissions allow an authorized user to create and control access to the Named Objects:

- Tables,
- Views,
- Procedures, and
- Functions.

This section describes permissions on tables, view, procedures, and functions. A user who creates an object is the owner of the object. The owner of the object or DBA can grant permissions using the GRANT command.

6.2.1 Authorities

There are two authorities in ASA: DBA and Resource. A user with DBA authority is exempt from Discretionary Access Control. This is the only type of trusted administrator within ASA. When a database is created, ASA creates a single usable user ID: DBA with DBA authority. However, the trusted administrator should use this account to create individual user accounts for each individual who should have DBA authority, so that there is individual accountability. The DBA can create database objects and assign ownership of the objects to other users. Only DBA can grant Resource authority. Resource authority is the permission to create database objects such as tables, views, stored procedures and triggers.

6.2.2 Tables

Any user with Resource authority can create a table. The permissions that can be granted on tables are as follows:

- ***SELECT*:** The *SELECT* permission allows users to read data from a table or a subset of a table, i.e., a set of columns. A user who has been granted the *SELECT* permission can execute the SELECT statement.

- ***INSERT***: The *INSERT* permission allows a user to insert rows into a table. A user who has been granted the *INSERT* permission can execute the INSERT statement.

- ***UPDATE:*** The *UPDATE* permission allows a user to update a row in a table. *UPDATE* may be granted on a whole table or a subset of the table, i.e., a set of columns. A user who has been granted the *UPDATE* permission can execute the UPDATE statement. If the UPDATE

statement has a WHERE clause, the *SELECT* permission is also required in addition to the *UPDATE* permission.

- ***DELETE:*** The *DELETE* permission allows a user to delete rows from a table. A user who has been granted the *DELETE* permission can execute the DELETE statement.

- ***ALTER:*** The *ALTER* permission allows a user to alter table structure and create triggers on a table. A user who has been granted the *ALTER* permission can execute the following SQL statements:

  - ALTER TABLE,
  - ALTER TRIGGER,
  - CREATE TRIGGER,
  - DROP TRIGGER, and
  - TRUNCATE TABLE.

- ***REFERENCES*:** The *REFERENCES* permission allows a user to create indexes on a table and to create foreign keys that reference the table. A user who has been granted the *REFERENCES* permission can execute the following SQL statements:

  - CREATE INDEX, and
  - DROP INDEX

6.2.3 Triggers

Triggers are considered table attributes and they are protected by the access controls on the table. In order to create a trigger, a user must have resource authority and either be the owner of the table or have *ALTER* permission on the table. When a trigger fires, it executes with the permissions of the owner of the table on which the trigger is defined. For example, if there is a DELETE trigger on Table A owned by user A and the trigger does a DELETE operation on Table B owned by User B, User A must have been granted *DELETE* permission on Table B.

A trigger on one table can fire a trigger on another table. For example, an INSERT operation in Table A can fire an INSERT trigger that results in an insert in Table B. Then the INSERT operation in Table B can fire an INSERT trigger that results in an INSERT in Table C. Finally, the INSERT in Table C can fire an INSERT trigger that results in an INSERT in Table D. Assuming Table A is owned by User A, Table B is owned by User B, Table C is owned by User C, and Table D is owned by User D, in order for the above scenario to work, User A must have been granted *INSERT* permission on Table B, User B must have been granted *INSERT* permission on Table C, and User C must have been granted *INSERT* permission on Table D.

The original INSERT and the resulting triggers are an atomic operation. Either the original INSERT and all the resulting triggers will succeed or none of them will. The whole operation will be rolled back. That is why triggers can be used to enforce integrity contraints. In the example above, if

INSERT permission on Table D were revoked from User C, the whole operation would be rolled back including the original INSERT into Table A.

6.2.4 Views

A user can create a view on a table if the user has *SELECT* permission on the base table(s). The permissions that can be granted on views are as follows: *SELECT, INSERT, UPDATE* and *DELETE.* The permissions on views behave similarly to the permissions on tables. The owner of a view can grant access to a view if the owner of the view has been granted the access with the GRANT option by the owner of the base table. Views are virtual objects. An update on a view results in the underlying base tables being updated.

6.2.5 Procedures and Functions

The only permission on procedures and functions is *EXECUTE*. A procedure executes with the permissions of its owner, not the calling user. If a procedure selects from Table A, the owner of the procedure must have *SELECT* permission on Table A. It is not necessary for the caller of the procedure to have *SELECT* permission on Table A. In this way, procedures can be used to limit a user's access to a table to certain pre-defined operations. DAC checks are performed when the procedure is compiled.

Safeguards have been implemented to ensure that the permission checks are current when a procedure is executed. First, whenever a GRANT or REVOKE is executed, all the permission lists associated with connection structures are cleared and recreated. The developer felt this was more efficient than attempting to insert or delete entries in the existing permission lists. Since grants and revokes are relatively rare events in a production environment, they feel the impact on performance is minimal. Second, an option has been implemented to require that procedures be recompiled when a grant or revoke is executed. This is necessary since permissions on objects accessed by the procedure are checked when the procedure is compiled. If a procedure owner's permission is revoked from a table, a user connecting after the revoke is executed should not be able to successfully execute the procedure.

A procedure is always executed with the permissions and authorities of its owner, irregardless of the permissions and authorities of the caller. Even if a user with DBA authority executes a procedure created by a user without DBA authority, the procedure will not execute with DBA authority. ASA maintains a stack of procedure calls to support nested procedure calls.

The access control policy for ASA is to control *EXECUTE* access on procedures and functions. *EXECUTE* permission is what allows a procedure to access data stored in user-defined tables. By default, procedure and function definitions stored in the SYSPROCEDURE system table are publicly accessible. The user data being protected is the data being accessed by the set of SQL statements, not the definition of the procedure itself.

6.2.6 Group Permissions

A group is a user ID granted the permission by DBA to have members. A group can also be a member of the group. Membership in a group can be granted by the DBA or by the group user ID. Any user ID can be a member of different groups. Permissions to the groups can be assigned in the

same way as individual users. These permissions can be inherited by members of the group including any other groups and their members. The DBA, Resource and GROUP permissions are not inherited by group members. The group members do not inherit the ownership of the database object associated with a user ID. Only granted permissions can be inherited by the group members.

6.2.7 Grant and Revoke Statements

The GRANT statement is used to grant database permissions to individual user IDs and groups. The GRANT statement is also used to create and delete users and groups.

- GRANT CONNECT allows a DBA to create a new user.
- GRANT DBA gives a user DBA authority.
- GRANT RESOURCE allows the user to create tables and views.
- GRANT GROUP allows the user to have group members.
- GRANT MEMBERSHIP IN GROUP allows the users to inherit table permissions from a group.
- GRANT SELECT, INSERT, UPDATE, or DELETE grants the specified permission on a table or view.
- GRANT ALTER or REFERENCES grants the specified permission on a table.
- GRANT ALL statement grants all the permissions on a table or view.
- GRANT WITH GRANT OPTION allows the named user ID to grant the same permissions to other users.
- GRANT EXECUTE ON can be used by DBA or owner of the procedure to grant execute permission to other users.
- GRANT INTEGRATED LOGIN TO allows the DBA to create an explicit integrated mapping between Windows NT user ID and existing database user ID.

The REVOKE statement is used to revoke permissions that were given using the GRANT statement. The permissions that can be revoked using REVOKE command are user permissions, table permissions and execute permissions on procedures. Except for the DBA, users can REVOKE only those permissions that they granted.

- REVOKE CONNECT is used to remove a user ID from a database.
- REVOKE GROUP will revoke group membership from all members of the group.
- REVOKE also allows the DBA or grantor of the permission to revoke *SELECT*, *INSERT*, *UPDATE*, *DELETE*, *ALTER* and *REFERENCES* permissions on the table from other users.
- REVOKE ALL statement revokes all the above permissions.
- REVOKE EXECUTE ON revokes execute permission on the procedure from the user.

Grants of DBA authority, Resource Authority, and *GROUP* permission take effect the next time that the user connects. DBA authority, Resource Authority, and *GROUP* permission cannot be revoked while a user is connected. If it is necessary to revoke these authorities or permissions immediately, a DBA can forcibly disconnect the user and then revoke the authority or permission. The revoke takes effect the next time the user connects.

Grant of table and view permissions, *SELECT, INSERT, UPDATE, DELETE, ALTER* and *REFERENCES,* take effect the next time that the user connects. Table and view permissions cannot be revoked from a user while the user is connected. If it is necessary to revoke these permissions immediately, a DBA can forcibly disconnect the user and then revoke the permission. The revoke takes effect the next time the user connects.

Grant of the *EXECUTE* permission on a procedure or function takes effect on the next statement. *EXECUTE* permission can be revoked while the user is connected. The revoke takes effect on the next statement.

Grants on tables and columns are independent. If a user is granted access to a table and a column within the table and revoked access on the table, the user will still have access to the column due to the Grant on the column. Similarly, permissions on all the columns but one in a table cannot be granted by granting the permission on the table and then revoking the permission on the one column. The permissions must be granted on the desired columns directly.

6.2.8 Changing Ownership on Nested Objects

Views and procedures can access underlying objects that are owned by different users. For example, if usera, userb, userc, and userd were four different users, userd.viewd could be based on userc.viewc, which could be based on userb.viewb, which could be based on usera.tablea. Similarly for procedures, userd.procd could call userc.procc, which could call userb.procb, which could insert into usera.tablea.

The following DAC rules apply to nested views and tables:

- In order to create a view, the user must have *SELECT* permission on all of the base objects (i.e., tables and views) in the view declaration.

- In order to access a view, the view owner must have been granted the appropriate permission on the underlying tables or views with the GRANT OPTION and the user must have been granted the appropriate permission on the view.

- Updating with a WHERE clause requires both *SELECT* and *UPDATE* permission.

- If a user owns the tables in a view definition, the user can access the tables through a view, even it the user is not the owner of the view and has not been granted access on the view.

The following DAC rules apply to nested procedures:

- A user does not require any permissions on the underlying objects (i.e., tables, views or procedures) in order to create a procedure.

- In order for a procedure to execute, the owner of the procedure needs the appropriate permissions on the objects referenced by the procedure.

- Even if a user owns all the tables referenced by a procedure, the user will not be able to execute the procedure to access the tables, unless the user has been granted *EXECUTE* permission on the procedure.

6.2.9 Permission-Related System Tables

The system tables related to discretionary access control are shown in Table 6.2.

Table 6.2 - System Tables Related to Permissions

System Table	Description
SYSCOLPERM	Contains columns with UPDATE permission
SYSGROUP	Describes many-to-many relationship between groups and members
SYSPROCPERM	Describes many-to-many relationship between procedures and users who have permission to call these procedures
SYSTABLEPERM	Contains permissions given by one user to other users using GRANT command
SYSUSERPERM	Contains description of user IDs such as user group, password, user name, authorities such as DBA authority
SYSDUMMY	This Dummy table can be used to find the current user ID

6.2.10 Implementation

A user structure has pointers to two lists: one for table and view permissions and the other for procedure and function permissions, which store the permissions that have been granted to the user. When a permission is granted or revoked to a user, the permission lists stored in memory are cleared and recreated from the system tables. (See Section 4.14 Query Processing.)

DAC checks are performed when a SQL statement is parsed.

6.3 Audit

The audit mechanism provided by ASA is independent of the NT audit mechanism. ASA generates two audit logs. The primary ASA audit log is the ASA transaction log file called <dbname>.log. Audit records generated while the engine is running are stored in the ASA transaction log. Audit records generated by database utilities that can execute when the engine is not running are stored in a separate database utility audit log file called <dbname>.alg.

This section addresses the following topics:

- Auditable Events,
- Audit Records,
- Audit Log Generation,
- Protection of Audit Data,
- Audit Log Management,
- Audit Reduction,

- Audit Correlation, and
- Audit Data Loss.

6.3.1 Auditable Events

The following activities are auditable:

- All successful and failed attempts to actions that require DBA access (such as creating users and setting public options)
- Any other information required to audit DBA actions
- All successful logins and failed login attempts
- All successful and failed DDL and DML statements
- All permissions checks

The types of auditable events in ASA are as follows:

- AUDIT_ENABLE_AUDITING - Auditing has been enabled or disabled.
- AUDIT_SET_PUB_OPTION - A public option has been set.
- AUDIT_OP_ATTEMPT - A DDL operation is being attempted.
- AUDIT_OP_SUCCESS - The previous DDL operation for this connection has finished (success or failure is indicated).
- AUDIT_PERM_CHECK - A permission check has been done (includes DBA/Resource authority, as well as object/column permissions).
- AUDIT_USER_CHECK - A user check has been done to determine ownership of objects (i.e. is current user the same as user X).
- AUDIT_CONNECTION - A connection attempt has occurred.
- AUDIT_SETUSER - A setuser command has been attempted.
- AUDIT_TRIGGER – A trigger has fired, or has finished executing.
- AUDIT_STRING – An arbitrary string is inserted into the audit trail by use of the sa_audit_string stored procedure. Execution of this stored procedure requires DBA authority.

When a user connects to the database, the connection is audited. The audit record for this connection includes the user identifier, a connection identifier, the byte offset of the record in the transaction log, and the date and time of the connection. The connection identifier is used in all subsequent audit records for this connection. Only the connection record has the user identifier. Disconnects are not logged.

The SQL statements for which transaction log entries are made include the following:

- SELECT, INSERT, UPDATE, and DELETE;
- ALTER DATABASE | DBSPACE | PROCEDURE | SERVER | TABLE | TRIGGER | VIEW | WRITEFILE;
- CHECKPOINT;
- COMMENT ON;

- CREATE COMPRESSED DATABASE | DATABASE | DBSPACE | EXPANDED DATABASE | EXTERNAL LOGIN | INDEX | MESSAGE | PROCEDURE | SCHEMA | SERVER | TABLE | TEMP TABLE | TRIGGER | TYPE | VIEW | WRITEFILE;
- DROP CONNECTION | DATABASE | DBSPACE | EXTERNAL LOGIN | INDEX | MESSAGE | OPTIMIZER STATISTICS | PROCEDURE | SERVER | TABLE | TRIGGER | VIEW;
- GRANT;
- INSTALL JAVA;
- REMOVE JAVA; and
- REVOKE.

Three database utilities perform actions that must be audited, but they do not necessarily communicate with a running database engine. Therefore, they cannot depend on the engine to audit their actions. These three database utilities, **dblog**, **dbwrite**, and **dbtran,** check the database or transaction log on invocation to see if auditing is enabled. If so, these utilities audit their invocation by writing to the database utility audit log file.

6.3.1.1 Audit of Permission Checks

There are several types of permission checks that are audited. Listed below in Table 6.3 are the different types, along with the information that is audited for each one.

Table 6.3 – Audit of Permission Checks

Permission check	Examples	Information audited
User authority	Does user have "resource" authority?	Authority
Permission on object	Does user have *SELECT* permission on table T?	Perm type, object name
Permission on column of table	Does user have *SELECT* permission on column c of table T?	Perm type, table, column

Valid permission types are: *SELECT, UPDATE, INSERT, DELETE, ALTER, REFERENCES, EXECUTE, GRANT SELECT, GRANT UPDATE, GRANT INSERT, GRANT DELETE, GRANT ALTER, GRANT REFERENCES.*

Permissions checks are made before the operation requiring the check is performed. Therefore, in some cases, permission checks may be made and audited for an operation that is then not performed.

6.3.1.2 Audit of Triggers

In general, audit records are associated with the user connection. In the case of triggers, audit records will be generated at the start and end of a trigger so that the DBA can easily determine whether or not an operation was performed by a trigger.

In the example below User C inserts a row into Table T2 which causes an Insert trigger to fire. The trigger on Table T2 inserts into Table T1. The audit trail would be as follows:

```
User C logs on - connection X
```

```
Connection X: Attempting insert into dbo.T2
Connection X: Checking insert permission on dbo.T2 - OK
Connection X: Insert into dbo.T2
Connection X: Firing trigger t2trig on table dbo.T2
Connection X: Attempting to insert into userc.T1
Connection X: Checking insert permission on userc.T1 - OK
Connection X: Insert into userc.T1
Connection X: Operation succeeded
Connection X: Trigger t2trig finished.
Connection X: Operation succeeded
```

The table owner can be determined by examining the SQL statement that caused the trigger to fire. The SQL statement for the same connection immediately preceding the "Firing trigger" record will be an INSERT, UPDATE, or DELETE. The table name will be in the format "owner.table".

6.3.1.3 Audit of Stored Procedures

A DAC check is performed when a procedure is executed and this check is auditable. It generates an AUDIT_PERM_CHECK record as defined in Table 6.5 below. The permission type is *EXECUTE*. The owner of the stored procedure is identified by the object name in the audit record. In the case of nested procedures, the owner of the calling procedure is not explicitly identified, although it could be determined by examining the previous permission check records for the connection.

6.3.2 Audit Records

Information recorded for each auditable event includes:

- user (except for failed login attempts)
- terminal ID or machine ID (for login attempts only)
- date and time of event, with resolution down to at least $1/100^{th}$ second
- Sequence number of the audit record within the event
- Success/failure

Not all of this information may be in one audit record, but it is found in the collection of audit records for the connection. For example, only the connection audit record contains the user identifier. The connection identifier ties together all of the records for that user for that connection.

Object names are qualified by the owner. The format is owner.object name. For example, table T1 owned by user SYS is identified in the audit record as SYS.T1.

The audit record contains a fixed portion containing date/time, audit type, and other information in the format shown in Table 6.4 below. The remainder of the record varies depending on the audit type as shown in Table 6.5 below.

Table 6.4 – Format of Audit Records - Fixed

Type	Format
transaction redo header	1 byte

connection identifier	3 bytes
date / time	11 bytes 2 bytes - year (e.g. 1998) 1 byte - month (1-12) 1 byte - day (1-31) 1 byte - hour (0-23) 1 byte - minute (0-59) 1 byte - second (0-59) 4 bytes – microsecond (0-999999)
Audit type	1 byte

Table 6.5 – Format of Audit Records – Variable by Type

Type	Format
AUDIT_ENABLE_AUDITING	1 byte - 1 (auditing enabled) or 0 (auditing disabled)
AUDIT_SET_PUB_OPTION	2 bytes - length of following string (n) n bytes - option name
AUDIT_OP_ATTEMPT	2 bytes - length of following string (n) n bytes - SQL of attempted operation
AUDIT_OP_SUCCESS	1 byte - 1 (operation succeeded) or 0 (operation failed)
AUDIT_PERM_CHECK	1 byte - 1 (success) or 0 (failure) 2 bytes - length of following string (n) n bytes - permission type (e.g. *SELECT, UPDATE, EXECUTE,* etc.) 2 bytes - length of following string (n) n bytes - object (table, view, procedure, etc.) name 2 bytes - length of following string (n) n bytes - column name, if applicable
AUDIT_USER_CHECK	1 byte - 1 (success) or 0 (failure) 2 bytes - length of following string (n) n bytes - user name
AUDIT_CONNECTION	1 byte - 1 (success) or 0 (failure) 2 bytes - length of following string (n) n bytes - user name (if connection succeeded) 2 bytes - length of following string (n) n bytes - machine ID
AUDIT_SETUSER	2 bytes - length of following string (n) n bytes - user name
AUDIT_TRIGGER	2 bytes – length of following string (n) n bytes – name of trigger 3 bytes – “fired” or “finished”
AUDIT_STRING	2 bytes – length of following string (n) n bytes – variable text string

The format for records in the database utility audit log is a single line as shown in Table 6.6.

Table 6.6 - Format of Database Utility Audit Log Records

Date	Time	ASA User ID	NT User Name	ASA DB Utility

6.3.3 Audit Log Generation

Auditing is turned on and off by using the **auditing** public option. Public options are set using the SET OPTION statement, which requires DBA authority. When the **auditing** option is on, all auditable events are audited.

Audit records are created and stored in a log page in the cache. When this page is full or when a COMMIT or CHECKPOINT occurs, the page is written to the log file on disk.

There is no limit on the size of audit log files. Audit log files grow until the file is truncated or deleted or disk space runs out. If the disk fills up while executing a command that would generate a record to the transaction log file, a "Disk Full" message is generated and the transaction is rolled back. No more commands that generate audit records can be executed against the database, when the transaction log is full. Similarly, if the disk becomes full while executing a database utility that writes to the database utility audit log (*.alg file), the utility will fail. In order to continue, files must be deleted or the audit log truncated to free up disk space.

6.3.4 Protection of Audit Data

Audit records are stored in transaction log files called <dbname>.log and the database utility audit logs called <dbname>.alg. These audit log files are stored in the same directory as the database file called <dbname>.db. This directory is protected by NT DAC and is only accessible by user Sybase, the NT Administrator, and the system.

When passwords are created and updated through the GRANT CONNECT SQL statement, they are stored in the transaction log in encrypted format.

6.3.5 Audit Log Management

The transaction log file can be truncated, purged or backed up using the **dblog** or **dbbackup** utilities as the Sybase user. The TFM directs the administrator to back up the transaction log file, before truncating it. Truncation of the audit log results in an audit record of the truncation being stored as the first record in the new audit log.

The database utility audit log (*.alg) file is in ASCII format. It can be truncated by deleting records with a text editor such as Notepad. Truncation of audit records in the *.alg file can be audited using the NT audit mechanism to audit file access to the *.alg file.

6.3.6 Audit Reduction

The transaction log contains all changes to the database in the order that they occur as well as audit records. Inserts, updates, deletes, commits, rollbacks, and database schema changes are therefore all available as part of the SQL script for recovery. The transaction log is not human-readable. The Log Translation utility (**dbtran**) with the –g switch can be used to display audit records. To

generate audit events for a single user, the **dbtran** -u switch is used. The connection ID relates all the records for a single user in the transaction log.

The **dbtran** utility was designed to create a SQL script from the transaction log file. This SQL script can be used to recover the database if a catastrophic failure occurs. The audit records are written to the transaction log as comments so that they will not be used in the recovery script. In addition, use of the –g switch alters the order in which the contents of the transaction log are presented. The recovery script is presented in "commit order", while the audit trail is presented in the order in which it was written to the log. Because the audit trail output cannot be used for recovery, the audit trail is presented with all entries (both SQL script and audit records) marked as comments when the –g switch is used.

The database utility audit log file (*.alg) is in ASCII format. Notepad can be used to view the audit log. The grep utility can be used to retrieve all the records for a single user, since each audit record is a single line.

6.3.7 Audit Correlation

Integrated login is required in order to be able to correlate records in the NT operating system security log with the ASA DBMS audit logs. Integrated login implements a one-to-one mapping between each ASA User ID and a Windows NT account. This information is stored in the SYSLOGIN system table, which can be read by a user with DBA authority. The NT account name is stored in the NT audit records. The ASA User ID is stored in the AUDIT_CONNECTION record along with the connection ID in the ASA transaction log. All subsequent audit records for the same user have the same Connection ID and can be related to the AUDIT_CONNECTION record. All of the records in the database utilities audit log (*.alg file) have the ASA User ID and the NT account name. Audit records in all three locations, the NT security log, the ASA transaction log, and the ASA database utility audit log, have time stamps.

In order to correlate records from more than one database, the audit logs from each database must be correlated with the NT audit log and then the resulting files merged.

6.3.8 Audit Data Loss

Some audit records can be lost before being written to permanent storage in case of failure of the ASA server or OS. The potential loss is equal to the size of the page in the cache. The size is dependent on how the database is created and may be set between one and 32 KB. The number of audit records that can be lost depends on the page size and other transaction log information for the SQL recovery script placed on the page.

In the worst case scenario, the page size is 32KB, the page is almost full when server dies and the page is full of audit records. The page is full of audit records when every audit record is for a SELECT statement, since any INSERT, UPDATE, DELETE, or DDL statement would be put in the transaction log. The shortest permission check for a select is for DBA authority. The variable audit record for that check is 11 bytes and 16 bytes for the fixed portion per record. So the maximum number of audit records that can be lost is 32K / (16+11) or approximately 1200 audit records.

6.4 Object Reuse

The ASA engine protects objects that are reused so subsequent subjects do not have access to data that was previously contained in an object.

The only storage object that is visible at the TCB interface is a row. When a row is deleted, a flag is set in the row descriptor that marks the row as "deleted", but does not actually delete or overwrite any data. Row operations will ignore rows marked as deleted, so that no user can see or reference the data contained within a deleted row. If the transaction that deleted the row is rolled back, the row is simply marked as "non-deleted" again, and the data becomes viewable again. If the transaction is committed, the data from any preceding rows is moved down within the page, overwriting the data from the deleted row. The number of rows on that page and the beginning of the free space are also updated within the page headers.

In addition to rows, ASA enforces object reuse on many of its internal storage objects as described below.

When pages are allocated, they are not zeroed. However, each page contains header information, which is initialized such that the length of the data within the page is set to 0. The engine code uses the length field of the page header to determine how much of the data within the page is usable, and data beyond this length is never referenced. These mechanisms ensure that any data that may already be in the page when it is allocated is not accessible. This is also true for streams buffers.

Most internal structures and objects, including those for containing table, view, procedure, and variable information, are C++ objects that inherit from a class called ZeroedHeapObject. This class overrides the standard new and delete operators, so that whenever objects that inherit from this class are created, they use an internal function called DV_Alloc to allocate the memory from the cache, and then the memory is zeroed. When these objects are deleted, the DV_Free function is called to return the memory to the cache. Memory is not zeroed on deletion.

7 Assurances

This section describes how ASA 7.0.0 with C2 Update satisfies the C2 System Architecture and Security Testing requirements.

7.1 System Architecture

This section discusses the mechanisms used by ASA to protect itself from unauthorized modification. It also describes mechanisms used by the TCB to ensure that information for one user is not passed to another user.

7.1.1 TCB Self-Protection

The composite TCB is composed the ASA and NT TCB subsets. ASA is installed as an NT service in the evaluated configuration. ASA runs under an NT account called "Sybase." The Sybase NT account must be granted the "Logon as a service" user right. This user right is checked when the ASA service is started up.

The ASA subset runs in user mode. The NT kernel runs in kernel mode so that ASA cannot tamper with the NT kernel when it is executing. Both the NT and the ASA subsets rely on the NT process isolation mechanism. ASA executing in user mode is protected by NT DAC mechanisms from other user mode processes.

ASA relies on the NT DAC mechanisms to protect its executable and database files. Permissions on the directory containing executable files are set to Full Access for the Sybase user and the NT administrator only. Untrusted users have read access. Database files, which include both TCB data and user data are stored in a protected directory. The permissions on this directory are set so that only the Sybase user, the NT administrator, and the SYSTEM have access. Untrusted users can only access TCB and user data through ASA; they have no access to ASA data through the operating system or other applications.

ASA stores its TCB data in protected system tables. The system tables are owned by the database user, DBO. DBO has a NULL password, and therefore, users cannot connect to the database as DBO. System tables cannot be modified by INSERT, UPDATE, or DELETE SQL statements. Each root database file contains a copy of the system tables. System tables are stored in database files that are protected by NT DAC, as stated above. The connection between a client application such as Interactive SQL and ASA is made through NT named pipes.

ASA controls access to the following named objects: tables, views, procedures, and functions. ASA isolates the resources to be protected so that they are subject to access control and auditing. This is done by ensuring the ASA resources can be accessed only through ASA application programming interfaces (APIs). These interfaces, including the Java API, allow SQL statements to be passed from the client to ASA. When SQL statements are parsed, ASA checks the client connection's user authorities and permissions before executing the statements. If the client connection does not pass

these checks, audit records are generated that indicate that the lack of an authority or permission prevented the requested access to the object.

ASA relies on the self-protection mechanisms of its ASA Java Virtual Machine (JVM) implementation to protect the TCB from Java applications. ASA JVMs protect themselves and offer additional protections from Java applications by implementing what is called a sandbox security model. The sandbox model limits certain actions that Java applications can perform (e.g., reading and writing to the local disk). ASA JVM sandboxes are maintained by a collection of components within the virtual machine, including the mechanism that loads classes, the mechanism that verifies class file formats, runtime mechanisms that control access to memory, and mechanisms that control access to outside of the ASA JVM. More details about the ASA JVM sandbox implementation can be found in Section 4.16.3, ASA JVM Self-Protection.

7.1.2 Task Isolation

ASA executes all requests on behalf of client connections, including requests that originate from the Java API. When a connection becomes active, the connection goes on a queue of connections that need servicing. If the Java API is invoked after a connection has been established, a new instance of the ASA Java Virtual Machine (JVM) is loaded into its own separate heap to maintain separate domains for each client connection. A request processing task sits in a loop selecting a connection that needs servicing and handling all of its pending requests. The request processing task has its own task stack. This stack is allocated by NT when the task is created by ASA and is protected by NT as part of the ASA process. When a request is received, it is placed in the request queue. When a request becomes active, a task descriptor is placed on a free request task stack. When the request servicing thread is ready to process the request, it takes the task descriptor off the stack and executes on behalf of the task. The task stack is used for any task-specific memory required to process the task.

7.2 Testing

7.2.1 Security Test Documentation

Sybase has developed security test documentation for its C2 evaluation in accordance with the Process Action Team (PAT) Guidance Working Group (PGWG) *Form and Content of Vendor Test Documentation* document. The ASA Test Documentation consists of the following documents:

- *Adaptive Server Anywhere C2 Test Plan.* The C2 Test Plan identifies all of the security relevant TCB interfaces and contains tests assertions for the security functionality at each TCB interface.

- *Adaptive Server Anywhere C2 Test Matrices.* The C2 Test Matrices map the test assertions to the applicable test suites.

- *Adaptive Server Anywhere C2 Test Procedures.* The C2 Test Procedures provide instructions on setting up the test environment, executing the Sybase ASA test suites, and interpreting the results.

- *Adaptive Server Anywhere C2 Test Procedures [Integrated Login]*. The C2 Test Procedures provide instructions on setting up the test environment, executing the Sybase ASA test suites, and interpreting the results for the integrated login tests.

- *DBTEST Documentation.* The DBTEST Documentation describes Sybase's automated test tools.

- Test suites and results.

7.2.2 ASA Test Suites

The Sybase security tests for ASA consisted of five test suites as follows:

- SQL: Tests SQL interfaces for Identification and Authentication, Discretionary Access Control, and Object Reuse.

- Java: Tests Java interfaces for Identification and Authentication, Discretionary Access Control, and System Architecture

- DB Tools: Tests database utilities that are executed from the command line for Identification and Authentication, and Discretionary Access Control.

- Application Programming Interfaces: Tests the DBLIB, ESQL, and NT 4.0 named pipes interfaces for Identification and Authentication, Discretionary Access Control, and System Architecture

- Audit: Tests all interfaces for audit.

These test suites were executed twice: once for the stand-alone engine and once for the network server, both of which were evaluated.

7.2.3 Evaluation Team Security Testing

The evaluation team conducted security testing of ASA at Sybase's facility in Waterloo, Ontario. The evaluation team installed Windows NT 4.0 and ASA 7.0.0 with C2 Update in their evaluated configurations. They then ran Sybase's security test suites and checked the actual results against the expected results. In addition, the evaluation team executed team tests documented in the Team Test Plan.

ASA successfully passed security testing without requiring any changes to its code. However, changes were required in the following areas:

- Integrated login is required in the evaluated configuration in order to meet the TDI audit correlation requirement.

- db_delete_file cannot be used in the evaluated configuration since the name of the deleted file is not included in the audit record that is generated when db_delete_file is executed.
- Changes to the test documentation and test suites.
- Several updates to the Trusted Facility Manual.
- Minor updates to the Security Features Users Guide.
- Minor updates to the design documentation.

8 Evaluation as a C2 Product

This chapter describes how the TCSEC requirements and the TDI interpretations are satisfied by Adaptive Server Anywhere 7.0.0 with C2 Update executing on Microsoft Windows NT 4.0 in NT's evaluated configuration. The rating earned by Sybase will be associated with the composite TCB comprising ASA and Windows NT. Consequently, requirements are considered from the perspective of the composite TCB. As described throughout this report, the ASA depends heavily on the operating system's security mechanisms to enforce its portion of the composite security policy; therefore, explanation of how some requirements are met lies in how the evaluated OS meets requirements. This chapter does not address the details of how the evaluated OS meets TCSEC requirements, but addresses the OS functions that ASA relies on to enforce its portion of the policy. For details concerning the OS design and implementation, see the Microsoft Windows NT 4.0 Final Evaluation Report (FER).

8.1 Discretionary Access Control

Requirement

TCSEC:

The TCB shall define and control access between named users and named objects (e.g., files and programs) in the ADP system. The enforcement mechanism (e.g., self/group/public controls, access control lists) shall allow users to specify and control sharing of those objects by named individuals, or defined groups of individuals, or by both, and shall provide controls to limit propagation of access rights. The discretionary access control mechanism shall, either by explicit user action or by default, provide that objects are protected from unauthorized access. These access controls shall be capable of including or excluding access to the granularity of a single user. Access permission to an object by users not already possessing access permission shall only be assigned by authorized users.

TDI:

The discretionary access control requirements apply as stated in the TCSEC to every TCB subset whose policy includes discretionary access control of its subjects to its objects. Any TCB subset whose policy does not include such discretionary access control is exempt from this requirement.[8]

Applicable Features

ASA enforces a Discretionary Access Control (DAC) policy that controls access between named users and named objects: tables, views, procedures, and functions. Object owners may grant permissions to individual users, or groups with the grant statement. Permissions may be granted with or without the grant option, which allows the grantee to propagate the permission to other users.

[8] Note that any evaluation by parts requires that at least one TCB subset in the TCB enforce a discretionary access control policy, and thus satisfy this requirement.

Permissions may be removed with the revoke statement. When objects are created, the default access is owner only access. Permissions may be granted to or revoked from a single user. ASA ensures that only object owners and users who have been granted a permission with the grant option are allowed to grant the permission to other users.

Conclusion

ASA satisfies the C2 Discretionary Access Control requirement.

8.2 Object Reuse

Requirement

TCSEC:

All authorizations to the information contained within a storage object shall be revoked prior to initial assignment, allocation or reallocation to a subject from the TCB's pool of unused storage objects. No information, including encrypted representations of information, produced by a prior subject's actions is to be available to any subject that obtains access to an object that has been released back to the system.

TDI:

This requirement applies as stated in the TCSEC to every TCB subset in the TCB.

Applicable Features

The ASA engine protects storage objects so information produced by a prior subject's actions is not available to subsequent subjects. The only storage object that is visible at the TCB interface is a row. When a row is deleted, a flag is set in the row descriptor that marks the row as "deleted", but does not actually delete or overwrite any data. Row operations will ignore rows marked as deleted, so that no user can see or reference the data contained within a deleted row. If the transaction that deleted the row is rolled back, the row is simply marked as "non-deleted" again, and the data is again viewable. If the transaction is committed, the data from any preceding rows is moved down within the page, overwriting the data from the deleted row. The number of rows on that page and the beginning of the free space are also updated within the page headers.

In addition to rows, ASA also enforces object reuse of many of its internal objects. For internal pages and streams buffers, ASA implements write before read. For internal heap objects, the memory is cleared before it is allocated.

Conclusion

ASA satisfies the C2 Object Reuse requirement.

8.3 Identification and Authentication

Requirement

TCSEC:

The TCB shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate. Furthermore, the TCB shall use a protected mechanism (e.g., passwords) to authenticate the user's identity. The TCB shall protect authentication data so that it cannot be accessed by any unauthorized user. The TCB shall be able to enforce individual accountability by providing the capability to uniquely identify each individual ADP system user. The TCB shall also provide the capability of associating this identity with all auditable actions taken by that individual.

TDI:

This requirement applies as stated in the TCSEC to the entire TCB. The cooperative action of the TCB subsets making up the TCB must satisfy the requirement.

If the TCB is composed of TCB subsets, one TCB subset may either rely on a mechanism in another TCB subset to provide identification and authentication services or provide the services directly. Each TCB subset may maintain its own identification and authentication data or one central repository may be maintained. If each TCB subset has its own data, then the information for each individual user must be consistent among all subsets.

Applicable Features

ASA limits access to users who have already passed authentication by the NT TCB subset and have established an NT session. From this authenticated NT session, ASA requires integrated login. This I&A feature relies on NT to authenticate the user; no separate password is stored by ASA; only a mapping must be established by the DBA between the NT username and an ASA database username. See Section 6.1.4, Integrated Login.

ASA also provides password protection for users with DBA authority, since they can bypass the integrated login mechanism, although the TFM directs them not to. See Section 6.1.5, Standard Login for more information about the password I&A mechanism.

The TCB protects authentication data so that it cannot be accessed by any unauthorized user. NT authentication data is protected by the NT TCB subset as described in the NT FER. ASA protects its authentication data by encrypting the password and storing it in a system table that is not accessible to untrusted users. See Section 6.1.1, System Tables and Section 6.1.6, Password Management.

The TCB is able to enforce individual accountability by providing the capability to uniquely identify each individual ADP system user. The TFM directs the DBA to assign each NT account and Sybase User ID to only one user.

The TCB provides the capability of associating this identity with all auditable actions taken by that individual. This capability is provided by the NT and ASA audit mechanisms. It is always possible to associate an audit record with the identity of the individual. See the NT FER and Section 6.3.3, Audit Records for details.

Conclusion

ASA satisfies the C2 Identification and Authentication requirement.

8.4 Audit

Requirement

TCSEC

TCB shall be able to create, maintain, and protect from modification or unauthorized access or destruction an audit trail of accesses to the objects it protects. The audit data shall be protected by the TCB so that read access to it is limited to those who are authorized for audit data. The TCB shall be able to record the following types of events: use of identification and authentication mechanisms, introduction of objects into a user's address space (e.g., file open, program initiation), deletion of objects, actions taken by computer operators and system administrators and/or system security officers, and other security relevant events. For each recorded event, the audit record shall identify: date and time of the event, user, type of event, and success or failure of the event. For identification/authentication events the origin of request (e.g., terminal ID) shall be included in the audit record. For events that introduce an object into a user's address space and for object deletion events the audit record shall include the name of the object. The ADP system administrator shall be able to selectively audit the actions of any one or more users based on individual identity.

TDI:

This requirement applies as stated in the TCSEC to the entire TCB. The cooperative action of the TCB subset making up the TCB must satisfy the requirement.

A TCB subset may maintain its own security audit log, distinct from that maintained by more primitive TCB subsets, or it may use an audit interface provided by a different TCB subset allowing the audit records it generates to be processed by that TCB subset.

If the TCB subset uses different user identifications than a more primitive TCB subset, there shall be a means to associate audit records generated by different TCB subsets for the same individual with each other, either at the time they are generated or later.

Auditable events, in the case of a database management system, are the individual operations initiated by untrusted users (e.g., SELECT, UPDATE, DELETE) not just the invocation of the database management system. The auditing mechanism shall have the capability of auditing all mediated accesses that are visible to users. That is, each discretionary access control policy decision shall be auditable. Individual operations performed by trusted software, if totally transparent to the

user, need not be auditable. If the total audit requirement is met by the use of more than one audit log, a method of correlation must be available.

Applicable Features

ASA maintains its own audit trail that is separate from the Windows NT operating system audit trail. ASA stores its audit records in its transaction log and the database utility audit log files. These files are protected from unauthorized access or modification by operating system DAC. Only the Database Administrator (DBA) has access to the transaction log through the **dbtran** utility.

ASA records security relevant events including login attempts, introduction of objects into the user's address space, deletion of objects, actions taken by the Database Administrator and other security relevant events as described in Section 6.3, Audit. The audit record records the date and time of the event, the user or connection ID, the type of event, and the success or failure of the event.

The ADP system administrator can use post-processing to selectively audit the actions of any one or more users based on individual identity. The DBA can select the events of a single user from the transaction log file using the -u option of the **dbtran** utility. The DBA can select the events of a single user from the database utility audit log file using the Notepad editor or the grep utility.

ASA uses integrated login to map NT accounts to ASA User IDs to meet the TDI audit correlation requirement.

Conclusion

ASA satisfies the C2 Audit requirement.

8.5 System Architecture

Requirement

TCSEC:

The TCB shall maintain a domain for its own execution that protects it from external interference or tampering (e.g., by modification of its code or data structures). Resources controlled by the TCB may be a defined subset of the subjects and objects in the ADP system. The TCB shall isolate the resources to be protected so that they are subject to the access control and auditing requirements.

TDI:

This requirement applies as stated in the TCSEC to every TCB subset, with the following additional interpretations.

The TCB must provide domains for execution that are allocated to and used by TCB subsets according to the subset-domain condition for evaluation by parts. A most primitive TCB subset must provide domains for execution. A less primitive TCB subset must make use of domains

provided by a more primitive TCB subset. A less primitive TCB subset may provide further execution domains but is not required to do so.

If the TCB is composed of multiple TCB subsets, this requirement applies to each TCB subset.

Applicable Features

The composite TCB is composed of ASA and NT TCB subsets. The NT subset maintains a domain for its own execution that protects it from external interference and tampering from other NT applications. The NT subset provides process isolation that protects the ASA subset from external interference and tampering. In the evaluated configuration, the ASA subset runs under the Sybase account as a service. The ASA subset runs in user mode along with other NT server processes and untrusted applications. The NT kernel runs in kernel mode so ASA cannot tamper with the NT kernel when it is executing. Both the NT and the ASA subsets rely on the NT process isolation mechanism.

ASA relies on the NT DAC mechanisms to protect its executables and storage objects. Untrusted users have no access to TCB executable files or data. ASA stores its TCB data in protected system tables. Access to TCB data by untrusted users is always mediated by the ASA subset.

ASA controls access to the following named objects: tables, views, procedures, and functions. ASA isolates the resources to be protected so they are subject to access control and auditing. This is done by ensuring the ASA resources can be accessed only through one of the allowed interfaces. No matter which interface is used, SQL statements are passed from the client to the ASA server. When SQL statements are parsed, ASA checks user authorities and permissions before executing the statements.

ASA relies on the self-protection mechanisms of its ASA Java Virtual Machine (JVM) implementation to protect the TCB from Java applications. ASA JVMs protect themselves and offer additional protections from Java applications by implementing what is called a sandbox security model. More details about the ASA JVM sandbox implementation can be found in Section 4.16.3, ASA JVM Self-Protection. ASA also provides isolation between ASA Java Virtual Machines. A new instance of the ASA Java Virtual Machine (JVM) is loaded into its own separate heap to maintain separate domains for each client connection.

Conclusion

ASA satisfies the C2 System Architecture requirement.

8.6 System Integrity

Requirement

TCSEC:

Hardware and/or software features shall be provided that can be used to periodically validate the correct operation of the on-site hardware and firmware elements of the TCB.

TDI:

This requirement applies as stated in the TCSEC to every TCB subset that includes hardware or firmware. Any TCB subset that does not include hardware or firmware is exempt from this requirement.

Applicable Features

This requirement does not apply to the ASA TCB subset, since it does not include hardware or firmware. The requirement is satisfied by the NT operating system TCB subset.

Conclusion

ASA satisfies the C2 System Integrity requirement.

8.7 Security Testing

Requirement

TCSEC:

The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. Testing shall be done to assure that there are no obvious ways for an unauthorized user to bypass or otherwise defeat the security protection mechanisms of the TCB. Testing shall also include a search for obvious flaws that would allow violation of resource isolation, or that would permit unauthorized access to the audit or authentication data.

TDI:

This requirement applies as stated in the TCSEC to the entire TCB. If a TCB consists of TCB subsets meeting the conditions for evaluation by parts, the satisfaction of the requirements by each TCB subset satisfies the requirement for the entire TCB. Otherwise, security testing of the entire TCB must be performed (even if the results of testing the individual TCB subsets were available).

Applicable Features

The security mechanisms of the ADP system were tested and found to work as claimed in the system documentation. The evaluation team performed the following testing activities: test coverage analysis, installation of NT and ASA in their evaluated configuration, execution of the ASA security test suite, and development and execution of evaluation team tests. During test coverage analysis, the team found a small number of missing test cases and these were added to the test suite. The evaluation team conducted on-site security testing of ASA at Sybase's facility in Waterloo, Ontario. The evaluation team installed Windows NT and ASA in their evaluated configurations. Then, they ran Sybase's security test suites and checked the actual results against the expected results. In addition, the evaluation team executed team tests documented in the Team Test Plan. ASA successfully completed security testing.

Testing was done to assure that there are no obvious ways for an unauthorized user to bypass or otherwise defeat the security protection mechanisms of the TCB. The team checked that all of the security functionality was tested for all of the security relevant TCB interfaces. In addition, the team included 37 tests in their team test suite to check that there was no way to bypass or defeat the security mechanisms of the TCB. These tests included tests to check that TCB programs and data were protected as documented.

Testing by the evaluation team also included a search for obvious flaws that would allow violation of resource isolation, or that would permit unauthorized access to audit or authentication data. The team specifically checked that the audit logs were stored in a directory protected by NT DAC as documented. The team also checked that overflow of the audit log was handled appropriately. In addition, the team checked that the SYSUSERPERM system table where passwords are stored was not accessible to untrusted users and that the passwords were not human readable.

The ASA TCB subsets met the conditions for evaluation by parts. Security testing of the NT operating system TCB subset is described in the NT FER. The ASA evaluation team performed testing of the ASA DBMS TCB subset.

Conclusion

ASA satisfies the C2 Security Testing requirement.

8.8 Security Features User's Guide

Requirement

TCSEC:

A single summary, chapter, or manual in user documentation shall describe the protection mechanisms provided by the TCB, guidelines on their use, and how they interact with one another.

TDI:

This requirement applies as stated in the TCSEC to every TCB subset in the TCB. This collection of guides must include descriptions of every TCB subset in the TCB and explicit cross-references to other related user's guides to other TCB subsets, as required. In addition, interactions between mechanisms within different TCB subsets must be clearly described.

Applicable Features:

The Security Features Users Guide consists of the following documents:

1) Sybase, Inc., Adaptive Server™ Anywhere 7.0.0 C2 Supplement, July 2000.

2) Sybase, Inc., *Adaptive Server™ Anywhere 7.0.0 C2 Documentation Update*, July 2000.

3) Sybase, Inc., *Adaptive Server™ Anywhere User's Guide*, Version 7.0.0, Document ID: MC0057, March 2000.

4) Sybase, Inc., *Introducing SQL Anywhere Studio*, Version 7.0.0, Document ID: MC0055, March 2000.

5) Sybase, Inc., *Adaptive Server™ Anywhere Reference Manual*, Version 7.0.0, Document ID: MC0058, March 2000.

6) Sybase, Inc., *Adaptive Server™ Anywhere Programming Interfaces Guide*, Version 7.0.0, Document ID: MC0059, March 2000.

7) Sybase, Inc., *Replication and Synchronization Guide, Chapter 26 – Command Reference for Adaptive Server Enterprise*, Version 7.0.0, Document ID: MC0061, March 2000.

The above documents work together to satisfy the SFUG requirements as follows:

1) The *C2 Supplement* provides guidance on integrated logins and connecting to ASA in the evaluated configuration that applies to non-administrative users. This document also provides warnings to users. This document is available on the Sybase support website as described in Appendix A.

2) The *C2 Documentation Update* contains security relevant changes to the 7.0.0 versions of the Reference Manual and User's Guide that will be included in future releases of ASA documentation. This document is available on the Sybase support website as described in Appendix A.

3) The *User's Guide* provides guidance on the use of the DAC and I&A mechanisms.

   Chapter 2 describes how to use integrated login.

   Chapter 23, "Managing User IDs and Permissions," provides a description of the available security features, guidelines on their use, and how they interact with one another. The first topic, "Database permissions overview," describes the administrative and ownership roles that exist within ASA (DBA Authority and Resource Authority), ownership permissions on tables and views, and the concept of group permissions. The second topic, "Managing individual user IDs and permissions," describes how to grant permissions on tables and views, grant users the right to grant permissions, grant EXECUTE permissions on procedures and triggers, and revoke permissions. The fourth topic, "Using views and procedures for extra security," describes how these mechanisms can interact to improve security. The fifth topic, "How user permissions are assessed," describes exactly how ASA determines the permissions of a user to a database object based on the interaction of the user's authorities, group memberships, and individual permissions. The last topic, "Users and permissions in the system tables," describes the system tables and the user's default permissions to access them.

4) *Introducing SQL Anywhere Studio* describes how to connect (login) to ASA from different client applications in Chapter 3.

5) The *Reference Manual* documents the functionality and permissions of many of the TCB interfaces such SQL statements, SQL functions, and database utilities.

6) The *Programming Interfaces Guide* describes the application programming interfaces.

7) The *Replication and Synchronization Guide* describes the ASE-compatible system stored procedures in Chapter 26.

Conclusion:

ASA satisfies the C2 Security Features User's Guide requirement.

8.9 Trusted Facility Manual

Requirement

TCSEC:

A manual addressed to the ADP system administrator shall present cautions about functions and privileges that should be controlled when running a secure facility. The procedures for examining and maintaining the audit files as well as the detailed audit record structure for each type of audit event shall be given.

TDI:

This requirement applies as stated in the TCSEC to the TCB and to every TCB subset in the TCB.

This requirement can be met by providing a set of manuals, one for each distinct (non-replicated) TCB subset. Each manual shall address the functions and privileges to be controlled for the associated TCB subset. Additionally, it must clearly show the interfaces to, and the interaction with, more primitive TCB subsets. The manual for each TCB subset shall identify the functions and privileges of the TCB subsets on which the associated TCB subset depends. Also, the TCB subset's manual shall identify which, if any, configuration options of the more primitive TCB subsets are to be used for the composite TCB to operate securely.

Any pre-defined roles supported (e.g., database administrator) by the TCB subset shall be addressed.

The means for correlating multiple audit logs and synonymous user identifications from multiple TCB subsets (if such exist) shall also be addressed.

The trusted facility manual shall describe the composite TCB so that the interactions among the TCB subsets can be readily determined. Other manuals may be referenced in this determination. The manuals that address the distinct modules of the TCB and the generation of the TCB need to be

integrated with the other trusted facility manuals only to the extent that they are affected by the use and operation (versus the development) of the composite TCB.

Applicable Features

The Trusted Facility Manual consists of the following documents:

1) Sybase, Inc., Adaptive Server™ Anywhere 7.0.0 C2 Supplement, July 2000.

2) Sybase, Inc., *Adaptive Server™ Anywhere 7.0.0 C2 Documentation Update*, July 2000.

3) Sybase, Inc., *Adaptive Server™ Anywhere User's Guide*, Version 7.0.0, Document ID: MC0057, March 2000.

4) Sybase, Inc., *Introducing SQL Anywhere Studio*, Version 7.0.0, Document ID: MC0055, March 2000.

5) Sybase, Inc., *Adaptive Server™ Anywhere Reference Manual*, Version 7.0.0, Document ID: MC0058, March 2000.

6) Sybase, Inc., *Adaptive Server™ Anywhere Programming Interfaces Guide*, Version 7.0.0, Document ID: MC0059, March 2000.

7) Sybase, Inc., *Replication and Synchronization Guide, Chapter 26 – Command Reference for Adaptive Server Enterprise*, Version 7.0.0, Document ID: MC0061, March 2000.

The above documents work together to satisfy the TFM requirements.

1) The *C2 Supplement* provides guidance on installation, configuration, auditing, restrictions in the evaluated configuration, security warnings, the C2 Patch, and where to look for more information. This document is the roadmap document for trusted administrators. The *C2 Supplement* describes how to install and configure the composite TCB. The *C2 Supplement* contains the procedures for examining, correlating, and maintaining the audit logs as well as the detailed audit record structure for each type of audit event. The *C2 Supplement* also contains an appendix that points to other document sections for utility and configuration parameter options.

   See the instructions in Appendix A for how to access this document on the Sybase web site.

2) The *C2 Documentation Update* contains security relevant changes to the 7.0.0 versions of the Reference Manual and User's Guide, which will be included in future releases of ASA documentation. This document is available on the Sybase website as described in Appendix A.

3) The *User's Guide* provides guidance on the configuration and use of the DAC and I&A mechanisms. Chapter 2 – Connecting to a Database and Chapter 23 - Managing User IDs and Permissions provide guidelines to operate ASA in a secure manner. They also provide guidelines to administrators for effective use of system's privileges and protection mechanisms.

Chapter 2 describes how to administer integrated login. It discusses security concerns caused by interaction of the protection mechanisms of the TCB subsets, specifically, enabling of the Guest account under NT and preventing blank passwords.

In Chapter 23, the first topic, Database Permissions Overview, describes the administrative and ownership roles that exist within ASA (DBA Authority and Resource Authority), ownership permissions on tables and views, and the concept of group permissions. The second topic, "Managing individual user IDs and permissions," describes how to create new users, change passwords, grant DBA and Resource authority, grant permissions on tables and views, grant users the right to grant permissions, grant EXECUTE permissions on procedures and triggers, and revoke permissions. The third topic, "Managing Groups," describes how to create groups, how to grant users membership in groups, and the special groups: SYS and PUBLIC. The fourth topic, "Using views and procedures for extra security," describes how these mechanisms can interact to improve security. The fifth topic, "How user permissions are assessed," describes exactly how ASA determines the permissions of a user to a database object based on the interaction of the user's authorities, group memberships, and individual permissions. The last topic, "Users and permissions in the system tables," describes the system tables and the user's default permissions to access them.

4) *Introducing SQL Anywhere Studio* describes how to connect (login) to ASA from different client applications in Chapter 3.

5) The *Reference Manual* documents the functionality and permissions of many of the TCB interfaces such as SQL statements, SQL functions, and database utilities. Chapter 2 provides guidance on starting the Database Server. Chapter 3 provides guidance on connecting. Chapter 4 provides guidance on the use of the database administration utilities. Chapter 5 provides guidance on configuring database options. Chapter 8 documents SQL statements. Chapter 9 provides a complete reference for SQL statements. It includes the permissions required to execute each SQL statement. It also provides guidance on how to use SQL statements such as the GRANT and REVOKE to manage the security functionality.

6) The *Programming Interfaces Guide* describes the application programming interfaces.

7) The *Replication and Synchronization Guide* describes the ASE-compatible system stored procedures in Chapter 26.

Conclusion

ASA satisfies the C2 Trusted Facility Manual requirement.

8.10 Test Documentation

Requirement

TCSEC:

The system developer shall provide to the evaluators a document that describes the test plan, test procedures that show how the security mechanisms were tested, and results of the security mechanisms' functional testing.

TDI:

This requirement applies as stated in the TCSEC to the composite TCB.

Applicable Features

The Sybase ASA Test Documentation consists of the following documents:

1. Sybase, Inc., *Adaptive Server Anywhere C2 Test Plan,*
2. Sybase, Inc., *Adaptive Server Anywhere C2 Test Matrices,*
3. Sybase, Inc., *Adaptive Server Anywhere C2 Test Procedures,*
4. Sybase, Inc., *Adaptive Server Anywhere C2 Test Procedures [Integrated Login],*
5. Sybase, Inc., *DBTEST Documentation,* and
6. Test suites and results.

Sybase developed security test documentation for its C2 evaluation in accordance with the Process Action Team (PAT) Guidance Working Group (PGWG) *Form and Content of Vendor Test Documentation* document. The main body of the C2 Test Plan is organized by security functionality. It contains assertions and pointers to lower level matrices. In addition, the Test Plan identifies interfaces that are not security relevant and do not need to be tested. The C2 Test Plan also has sections on the Testing Environment and Automated Tools. In the matrix document, each lower level matrix contains rows consisting of three cells: the name of the interface, a pointer to the relevant assertions in the C2 Test Plan, and a pointer to the associated test suite. The Test Procedures provide instructions on how to set up the test environment, execute the test suites, and interpret the results. Sybase provided copies of their security test suites and results for team to analyze during Test Coverage Analysis. The Assertions section of the Security Test Plan contains English language descriptions of the expected results. The test logs contain the expected results generated by the test software. The team reviewed both sets of results to ensure that they were consistent.

Conclusion

ASA satisfies the C2 Test Documentation requirement.

8.11 Design Documentation

Requirement

TCSEC:

Documentation shall be available that provides a description of the manufacturer's philosophy of protection and an explanation of how this philosophy is translated in to the TCB. If the TCB is composed of distinct modules, the interfaces between these modules shall be described.

TDI:

This requirement applies as stated in the TCSEC to the composite TCB. If the TCB is composed of multiple subsets, this requirement applies to each TCB subset and the interfaces between TCB subsets.

Applicable Features

Sybase provided the following materials to satisfy the design documentation requirement:

- User Documentation
  - *Adaptive Server™ Anywhere Programming Interfaces Guide*
  - *Adaptive Server™ Anywhere Reference Manual*
  - *Adaptive Server™ Anywhere User's Guide*
  - *First Guide to SQL Anywhere™ Studio*

- Sybase Fundamentals Training Materials

- Responses to Trusted Product Evaluation Questionnaire, NCSC-TG-019, Version-2

- Design Notes

  - Windows NT APIs
  - Windows NT DLLs
  - User Structure
  - Communication in ASA
  - Streams Interface
  - SQLPres Data Unit Types
  - SQLPres Command Sequences
  - Memory Usage
  - Tasking, Cache Manager, and Page Replacement
  - Object Reuse
  - Audit Design for ASA
  - System Stored Procedures
  - DBSVC command

- Header Files

The above documents work together to satisfy the C2 documentation requirement.

1) The user documentation identifies the TCB interfaces and describes the security semantics for each interface. They also describe the security functionality and how to manage it.

2) The training materials provide an overview of the system architecture.

3) The responses to the Trusted Product Evaluation Questionnaire describe the philosophy of protection.

4) The Design Notes provide detailed information on interfaces and implementation.

5) The header files provide information on TCB internal data structures.

Conclusion

ASA satisfies the C2 Design Documentation requirement.

9 TCSEC Interpretations

9.1 Audit

9.1.1 I-0004 Enforcement of audit settings consistent with protection goals

9.1.1.1 Text of Interpretation

This interpretation is effective 1993-10-20 and applies to classes C2, B1, B2, B3, and A1.

The following interprets the requirement that "The ADP system administrator shall be able to selectively audit . . ."

If the TCB supports the selection of events to be audited, it shall provide a method for immediate enforcement of a change in audit settings (e.g., to audit a specified user, to audit objects at a particular sensitivity level); however, the immediate method (e.g., shutting the system down) need not be the usual method for enforcement. The TFM shall describe both the usual enforcement of audit settings and, if the immediate enforcement method is different from the usual one, how an administrator can cause immediate enforcement. The TFM shall describe the consequences of changing an audit state dynamically if such changes could result in incomplete or misleading audit data.

9.1.1.2 Application to Adaptive Server Anywhere

The ASA TCB does not provide the ability to change to audit settings. If auditing is enabled, all auditable events are audited. Therefore, this interpretation is not applicable

Note that the requirement for the ADP administrator to be able to audit the events of a single user is satisfied by using either the **dbtran** utility or Notepad after the audit data has been collected. The –u option of the **dbtran** utility can be used to select the events of a single user in *.log files. Audit records generated when the engine is not running are stored in ASCII format in the *.alg file. The Notepad text editor can be used to select the records of a single user from a *.alg file.

9.1.2 I-0005 Action for audit log overflow

9.1.2.1 Text of Interpretation

This interpretation is effective 1993-10-20 and applies to classes C2, B1, B2, B3, and A1.

The following interprets the requirement that "The TCB shall be able to create, maintain and protect from modification or unauthorized access or destruction an audit trail of accesses to the objects it protects."

When the TCB becomes unable to collect audit data, it shall give a clear indication of this condition and take a pre-specified action. The system implementation may allow an administrator to choose from a range of actions. One of the actions that may be chosen by the administrator (or, if no choice is possible, the only action) shall be that the system cease performing auditable events when the TCB is unable to collect audit data. Choosing an audit overflow action shall be considered a security-relevant administrative event for the purposes of auditing. The TFM shall fully describe the administrator's options.

9.1.2.2 Application to Adaptive Server Anywhere

Audit records are stored in the transaction log. There is no limit on the size of audit log files. Audit log files grow until the file is truncated or deleted or disk space runs out. If the disk fills up while executing a command that would generate a record to the transaction log file, a "Disk Full" message is generated and the transaction is rolled back. No more commands that generate audit records can be executed against the database, when the transaction log is full. Similarly, if the disk becomes full while executing a database utility that writes to the database utility audit log (*.alg file), the utility will fail. In order to continue, files must be deleted or the audit log truncated to free up disk space.

The only possible action is for the DBA in the evaluated configuration is to backup the full transaction log and then truncate it. The DBA could also turn the transaction log off, but this would put the product outside of the evaluated configuration. When the transaction log is truncated, an audit record is generated that is stored in the new transaction log. The database utility audit log (*.alg) is an ASCII test file that can be truncated using a text editor such as Notepad. Truncation of the *.alg file can be audited by auditing access to the file using the NT audit mechanism.

9.1.3 I-0006 Audit of user-id for invalid login

9.1.3.1 Text of Interpretation

This interpretation is effective 1993-10-20 and applies to classes C2, B1, B2, B3, and A1.

The following interprets the requirement that "The TCB shall be able to record the following types of events: use of identification and authentication mechanisms, . . . For each recorded event, the audit record shall identify: date and time of the event, user, type of event, and success or failure of the event."

While the audit mechanism is required to be capable of producing a record of each login attempt, on failed login attempts it is not required to record in the audit record the character string supplied as the user identity.

9.1.3.2 Application to Adaptive Server Anywhere

For failed login attempts, ASA records the character string supplied for user identity.

9.1.4 I-0043 Auditing use of unnamed pipe

9.1.4.1 Text of Interpretation

This interpretation is effective 1994-04-19 and applies to classes C2, B1, B2, B3, and A1.

The following interprets the requirements that "The TCB shall be able to record the following types of events. . . : introduction of objects into a user's address space (e.g., file open, program initiation), . . . For events that introduce an object into a user's address space . . . "

In products that support mechanisms similar to unnamed pipes in UNIX systems, the creation of an unnamed pipe shall be auditable; however, the auditing may be delayed until the pipe becomes sharable with another subject (e.g., the creation of a child or the passing of the pipe's descriptor to another subject). At each point that the unnamed pipe is shared, the sharing must be auditable.

9.1.4.2 Application to Adaptive Server Anywhere

Not applicable. ASA does not implement unnamed pipes or other similar sharing mechanisms.

9.1.5 I-0073 OK to audit decision regardless of whether action completed

9.1.5.1 Text of Interpretation

This interpretation is effective 1993-10-20 and applies to classes C2, B1, B2, B3, and A1.

The following interprets the requirement that "The TCB shall be able to record the following types of events: . . . introduction of objects into a user's address space (e.g., file open, program initiation), . . ."

Auditing the attempted introduction of an object at the point of passing the security access control checks satisfies the above requirement, even though the object may not actually be introduced into the subject's address space because of failing later checks not related to security.

9.1.5.2 Application to Adaptive Server Anywhere

ASA generates audit records at the time that a permission check is made and after successful actions. It is possible for an action to fail for non-security relevant reasons, after the permission check is made. In this case, there will be an audit record generated, although the action failed.

9.1.6 I-0286 Auditing unadvertised TCB interfaces

9.1.6.1 Text of Interpretation

This interpretation is effective 1994-04-18 and applies to classes C2, B1, B2, B3, and A1.

The following interprets the requirement that "The TCB shall be able to record the following types of events . . . and other security-relevant events."

The TCB shall be capable of auditing all security-relevant events that can occur during the operation of the evaluated configuration, including events that may result from the use of TCB interfaces not advertised for general use.

9.1.6.2 Application to Adaptive Server Anywhere

Not applicable. There are no unadvertised TCB interfaces.

9.2 Configuration Management

No requirement at C2.

9.3 Covert Channel Analysis

No requirement at C2.

9.4 Design Documentation

9.4.1 I-0192 Interface manuals as design documentation

9.4.1.1 Text of Interpretation

This interpretation is effective 1994-04-19 and applies to classes C1, C2, B1, B2, B3, and A1.

The following interprets the requirement that "Documentation shall be available that provides a description of the manufacturer's philosophy of protection and an explanation of how this philosophy is translated into the TCB."

Interface-reference manuals (e.g., UNIX manual pages) are not sufficient, by themselves, as TCB design documentation.

9.4.1.2 Application to Adaptive Server Anywhere

The *ASA Reference Manual* describes its TCB interfaces. However, the interface descriptions include security relevant information such as the DAC checks performed (i.e., DBA authority, owner, and permission checks). In addition, the *Reference Manual* provides examples and scenarios of how the interface operates or does not operate as a result of those permission checks.

9.4.2 I-0193 Standard system books as design documentation

9.4.2.1 Text of Interpretation

This interpretation is effective 1993-10-20 and applies to classes C1, C2, B1, B2, B3, and A1.

The following interprets the requirement that "Documentation shall be available that provides a description of the manufacturer's philosophy of protection and an explanation of how this philosophy is translated into the TCB".

Books describing the design and implementation of a system used as the basis for a trusted system, but which are inaccurate or incomplete for the trusted system implementation, are not sufficient as design documentation. Such books may partially fulfill the requirement if additional documentation provides a complete description of all differences between the book's description and the actual system implementation, including a satisfactory description of any parts of the TCB not described in the book(s).

9.4.2.2 Application to Adaptive Server Anywhere

Where applicable, design notes and header files were provided to supplement the published documentation.

9.5 Design Specification and Verification

No requirement at C2.

9.6 Device Labels

No requirement at C2.

9.7 Discretionary Access Control

9.7.1 I-0002 Delayed revocation of DAC access

9.7.1.1 Text of Interpretation

This interpretation is effective 1993-10-20 and applies to classes C1, C2, B1, B2, B3, and A1.

The following interprets the requirement that "The TCB shall define and control access between named users and named objects (e.g., files and programs) in the ADP system."

A TCB is not required to provide any mechanism for the immediate revocation of DAC access to an object where access has already been established (e.g., opened) when access to that object is reduced. It is sufficient for the SFUG and other documentation to describe the product's revocation policy. However, a change in DAC permissions shall have an immediate effect on attempts to establish new access to that object.

9.7.1.2 Application to Adaptive Server Anywhere

The ASA implementation ensures that DAC permissions such as *SELECT, INSERT, UPDATE, DELETE, ALTER* and *REFERENCES* take effect on attempts to establish new access to objects. This is done by deleting all permission information from cache, whenever a grant or revoke command is issued in the evaluated configuration.

It is not possible to revoke DBA authority, Resource authority, or table permissions while a user is connected. However, a user can be forcibly disconnected by another user with DBA authority after which that user's DBA authority, Resource authority, or table permissions can then be revoked.

9.7.2 I-0020 DAC authority for assignment

9.7.2.1 Text of Interpretation

This interpretation is effective 1993-10-20 and applies to classes C1, C2, B1, B2, B3, and A1.

Starting at C1 the following interprets the requirement that "The enforcement mechanism . . . shall allow users to specify and control sharing . . ." At C2 it also interprets the requirement that "Access permission . . . shall only be assigned by authorized users."

A TCB need not provide all users with the capability to control the sharing of objects. A DAC policy where only system administrators assign access to objects can satisfy the DAC requirement. The SFUG shall clearly identify the roles or user types (e.g., system administrator) who can control sharing.

9.7.2.2 Application to Adaptive Server Anywhere

The owner of an object, not the administrator, controls access to the object. However, a trusted user with DBA authority can grant permissions on objects owned by other users.

The SFUG clearly identifies the roles or user types (e.g., system administrator) who can control sharing. Section 23 – Managing User IDs and Permissions of the *User's Guide* describes how the DBA and object owner can control sharing. In addition, there is a permissions section for each SQL statement in the *Reference Manual*.

9.7.3 I-0053 Public objects and DAC

9.7.3.1 Text of Interpretation

This interpretation is effective 1995-01-12 and applies to classes C1, C2, B1, B2, B3, and A1.

The following interprets the entire Discretionary Access Control requirement.

An object for which the TCB unconditionally permits all subjects "read" access shall be considered a public object, provided that only the TCB or privileged subjects may create, delete, or modify the object. No discretionary access checks or auditing are required for "read" accesses to such objects. Attempts to create, delete, or modify such objects shall be considered security- relevant events, and, therefore, controlled and auditable. Objects that all subjects can read must be, implicitly, system low.

9.7.3.2 Application to Adaptive Server Anywhere

Only the system or users with DBA authority can create, delete, or modify public objects.

ASA does not implement MAC or sensitivity labels, so system low is not applicable.

9.7.4 I-0222 Passwords not acceptable for DAC

9.7.4.1 Text of Interpretation

This interpretation is effective 1994-12-05 and applies to classes C1, C2, B1, B2, B3, and A1.

The following interprets the requirement that "The enforcement mechanism (e.g., self/group/public controls, access control lists) shall allow users to specify and control sharing of those objects by named individuals or defined groups or both."

The TCB shall not depend solely on passwords as an access control mechanism. If passwords are employed as part of an access control mechanism, they shall not be considered sufficient to satisfy any aspect of the DAC requirement.

9.7.4.2 Application to Adaptive Server Anywhere

ASA does not depend solely on passwords for DAC. Access to named objects is controlled by authorities and permissions granted to named users.

9.7.5 I-0312 Set-ID and the DAC requirement

9.7.5.1 Text of Interpretation

This interpretation is effective 1994-04-19 and applies to classes C1, C2, B1, B2, B3, and A1.

The following interprets the entire DAC requirement and applies to products that implement a mechanism similar to the set-user ID/set-group ID (set-ID) mechanism defined in IEEE Standard P1003.1.

The set-ID mechanism can be part of an acceptable DAC implementation.

9.7.5.2 Application to Adaptive Server Anywhere

ASA procedures or functions are equivalent to a set_ID mechanisms in that they execute with the permissions of the owner of the procedure.

In addition, there is a SET USER SQL statement that can only be executed by a user with DBA authority.

9.8 Exportation of Labeled Information

No requirement at C2.

9.9 Exportation to Multilevel Devices

No requirement at C2.

9.10 Exportation to Single-Level Devices

No requirement at C2.

9.11 Identification and Authentication

9.11.1 I-0001 Delayed enforcement of authorization change

9.11.1.1 Text of Interpretation

This interpretation is effective 1994-04-19 and applies to classes C1, C2, B1, B2, B3, and A1.

The following interprets the requirements that at C1 "The TCB shall protect authentication data so that it cannot be accessed by any unauthorized user." and at B1 "Furthermore, the TCB shall maintain . . . information for determining the clearance and authorizations of individual users. This data shall be used . . . to ensure that the security level and authorizations of subjects external to the TCB that may be created to act on behalf of the individual user are dominated by the clearance and authorization of that user."

If a TCB supports security-relevant authorizations then it shall provide a method for immediate enforcement of removing those authorizations. However, the immediate method (e.g., shutting the system down) need not be the usual method for enforcement. The TFM shall describe both the usual enforcement of granting and removing authorizations and, if the immediate enforcement method is different from the usual one, how an administrator can cause immediate enforcement.

9.11.1.2 Application to Adaptive Server Anywhere

ASA does not have any subject security attributes that are equivalent to a clearance level that are established at the time of a connection. DBA authority and Resource authority cannot be revoked from a connected user. However, another user with DBA authority can forcibly disconnect the user and then revoke DBA or Resource authority. The TFM describes how an administrator can cause immediate enforcement.

9.11.2 I-0096 Blanking passwords

9.11.2.1 Text of Interpretation

This interpretation is effective 1994-04-18 and applies to classes C1, C2, B1, B2, B3, and A1.

The following interprets the requirement that "The TCB shall protect authentication data so that it cannot be accessed by any unauthorized user."

The TCB shall not produce a visible display of any authentication data entered through the keyboard (e.g., by echoing).

9.11.2.2 Application to Adaptive Server Anywhere

Passwords are echoed on the “GRANT CONNECT TO ... IDENTIFIED BY ... statement. However, this is not a problem in the evaluated configuration. Integrated login is required in the evaluated configuration, so although a password must be entered it is not used for user authentication. DBAs can bypass the integrated login mechanism and login using their password, although the TFM directs them not to. DBAs must protect the DBA's passwords, since an attacker could login as the DBA if they knew their password. DBAs are directed to type in the GRANT CONNECT statement so that the password is not visible to untrusted users.

9.11.3 I-0240 Passwords may be used for card input

9.11.3.1 Text of Interpretation

This interpretation is effective 1993-10-20 and applies to classes C1, C2, B1, B2, B3, and A1.

The following interprets the requirement that "Furthermore, the TCB shall use a protected mechanism (e.g., passwords) to authenticate the user's identity. The TCB shall protect authentication data so that it cannot be accessed by any unauthorized user."

The card input of batch jobs may contain human-readable user passwords. The TFM and the SFUG for the product shall explain the risks in placing passwords on card input and shall suggest procedures to mitigate that risk.

9.11.3.2 Application to Adaptive Server Anywhere

Not applicable. ASA does not use card input.

9.11.4 I-0288 Actions allowed before I&A

9.11.4.1 Text of Interpretation

This interpretation is effective 1994-04-18 and applies to classes C1, C2, B1, B2, B3, and A1.

The following interprets the requirement that "The TCB shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate."
Prior to having been identified and authenticated by the TCB, a user communicating with the TCB may be allowed to perform only those actions that would not require TCB mediation.

9.11.4.2 Application to Adaptive Server Anywhere

Users must be identified before they can perform any actions that the TCB is expected to mediate. The only operation that can be performed by an unidentified user in ASA is to execute the db_find_engine command to locate a particular database engine.

9.12 Label Integrity

No requirement at C2.

9.13 Labeling Human-Readable Output

No requirement at C2.

9.14 Labels

No requirement at C2.

9.15 Mandatory Access Control

No requirement at C2.

9.16 Object Reuse

9.16.1 I-0041 Object reuse applies to all system resources

9.16.1.1 Text of Interpretation

This interpretation is effective 1994-04-19 and applies to classes C2, B1, B2, B3, and A1.

The following interprets all of the Object Reuse requirement.

Analysis for residual data shall be performed on all sharable objects and their attributes (i.e., objects to which MAC or DAC are applied) and other system resources (e.g., stacks, process memory).

9.16.1.2 Application to Adaptive Server Anywhere

The only sharable object visible at the TCB interface is a row. Analysis was performed on rows and the internal system objects such as memory pages, cache pages, and buffers where rows may be stored.

9.17 Security Features User's Guide

9.17.1 I-0244 Flexibility in packaging SFUG

9.17.1.1 Text of Interpretation

This interpretation is effective 1993-10-20 and applies to classes C1, C2, B1, B2, B3, and A1.

The following interprets the entire SFUG requirement.

All SFUG documentation shall be in a form that system administrators and users can read at the time that an understanding of the topics covered is needed to use the system in a secure manner (e.g., it shall not be required that the user login in order to read instructions about how to login). The documents or portions of documents that make up the SFUG shall be precisely identified. There are no further restrictions on the packaging (one document, several documents, parts of several documents) or delivery (hardcopy, online) of the SFUG.

9.17.1.2 Application to Adaptive Server Anywhere

The SFUG is contained in several documents. The Reference Manual, User's Guide and other documents are available on the installation CD. They can be accessed before ASA is installed and without requiring a database connection after ASA is installed. In addition, the C2 Supplement and the C2 Documentation Update are available on the Sybase website as described in Appendix A.

9.18 Security Testing

9.18.1 I-0170 Functional tests required for object reuse

9.18.1.1 Text of Interpretation

This interpretation is effective 1994-04-18 and applies to classes C2, B1, B2, B3, and A1.

The following interprets the requirement that "The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. Testing shall be done to assure that there are no obvious ways for an unauthorized user to bypass or otherwise defeat the security protection mechanisms of the TCB."

TCB interface(s) that allow manipulation and review of the contents of a subject's address space and of other resources available at the TCB interface (storage and named objects, devices) shall have functional tests included in the vendor test suite to supplement the analysis for object reuse.

9.18.1.2 Application to Adaptive Server Anywhere

Object reuse tests are included in ASA's security test suite.

9.19 Subject Sensitivity Labels

No requirement at C2.

9.20 System Architecture

9.20.1 I-0213 Administrator interface is part of TCB

9.20.1.1 Text of Interpretation

This interpretation is effective 1995-01-12 and applies to classes C2, B1, B2, B3, and A1.

The following interprets the requirement at C2 that "The TCB shall isolate the resources to be protected so that they are subject to the access control and auditing requirements."

Those components of the product that provide the interfaces required for performing administrative actions shall be considered TCB components. The "administrative actions" to which this interpretation applies shall be those that are defined in the TFM to be performed by administrative personnel (e.g., operators, system administrators, system security officers) while the product is in its secure operational state. The TFM shall clearly identify which mechanisms are, and which are not, acceptable for performing each administrative action.

9.20.1.2 Application to Adaptive Server Anywhere

The administrative interfaces were included in the TCB. They include the DBISQL interactive SQL tool, and the database utilities. The C2 Supplement provides guidance on which mechanisms are and are not acceptable for performing administrative actions in the evaluated configuration.

9.21 System Integrity

9.21.1 I-0144 Availability of diagnostics

9.21.1.1 Text of Interpretation

This interpretation is effective 1994-11-16 and applies to classes C1, C2, B1, B2, B3, and A1.

The following interprets the entire System Integrity requirement.

If the features provided by the vendor to meet this requirement cannot be exercised by the purchaser of the product, the vendor shall make available appropriate services to use the features as needed to meet the requirement. These services shall be available on an "on-demand" basis.

9.21.1.2 Application to Adaptive Server Anywhere

Not applicable. The system integrity requirement is satisfied by the Windows NT OS subset of the composite TCB. ASA does not have any hardware or firmware as part of its TCB subset.

9.22 Test Documentation

No interpretations.

9.23 Trusted Distribution

No requirement at C2.

9.24 Trusted Facility Management

No requirement at C2

9.25 Trusted Facility Manual

9.25.1 I-0046 Detailed audit record structure

9.25.1.1 Text of Interpretation

This interpretation is effective 1994-07-12 and applies to classes C2, B1, B2, B3, and A1.

The following interprets the requirement that "The procedures for examining and maintaining the audit files as well as the detailed audit record structure for each type of audit event shall be given."

The documentation of the detailed audit record structure may describe either the raw records produced by the audit mechanism or the output of an audit post-processing tool as long as the records described contain the information specified in the audit requirement. If the output of the audit post-processing tool is described, the tool is considered part of the TCB.

9.25.1.2 Application to Adaptive Server Anywhere

The DBA uses the **dbtran** post-processing tool to retrieve audit records in human readable form. The C2 Supplement and the FER describe the audit record structures and the latter document provides sample records.

9.25.2 I-0069 Flexibility in packaging TFM

9.25.2.1 Text of Interpretation

This interpretation is effective 1993-10-20 and applies to classes C1, C2, B1, B2, B3, and A1.

The following interprets the requirement that "A manual addressed to the ADP system administrator shall present cautions about functions and privileges that should be controlled when running a secure facility."

All TFM documentation shall be in a form that system administrators and users can read at the time that an understanding of the topics covered is needed to use the system in a secure manner (e.g., it shall not be required that the system be brought up in order to read instructions about how to bring up the system). The documents or portions of documents that make up the TFM shall be precisely identified. There are no further restrictions on the packaging (one document, several documents, parts of several documents) or delivery (hardcopy, online) of the TFM.

9.25.2.2 Application to Adaptive Server Anywhere

The TFM is contained in several documents including a Reference Manual and User's Guide that are formatted so that they can be accessed on-line. In addition, the C2 Supplement and C2 Documentation Update are available on-line at the Sybase web site. See Section 8.9, Trusted Facility Manual, for the list of documents that comprise the TFM. See Appendix A for how to access the C2 Supplement and C2 Documentation Update.

9.26 Trusted Path

No requirement at C2.

9.27 Trusted Recovery

No requirement at C2.

10 Appendix A - Evaluated Software Components

The evaluated software configuration is Sybase, Inc.'s Adaptive Server Anywhere 7.0.0 with C2 Update (ASA) running on Microsoft's Windows NT 4.0 with Service Pack 6a with C2 Update as configured by the *ASA 7.0.0 C2 Supplement*, a component of the ASA Trusted Facility Manual.

Except for the C2 Supplement and the C2 Documentation Update, the documents that comprise the SFUG and TFM are provided with the ASA 7.0.0 with C2 Update software in hardcopy and/or on the product CD. (See Section 8.8, Security Features User's Guide and Section 8.9, Trusted Facility Manual.) The C2 Supplement, the C2 Documentation Update, and the C2 Update/Patch required for product installation must be downloaded from the web site that Sybase has established for this product. That web site can be accessed as follows:

1. Go to http://www.sybase.com/developer
2. Click on "Mobile & Wireless Developer" in the list on the left
3. Click on "Downloads" in the list on the left
4. Follow the link in the resulting list to the page containing the C2 documents and C2 Patch.

The C2 Supplement provides detailed instructions for installing ASA in its evaluated configuration. In addition, it specifies the software functionality that is included in or excluded from the evaluated configuration. For convenience, the major features of the evaluated configuration from the C2 Supplement are summarized below:

- The ASA evaluated software includes both the stand-alone engine (dbeng7.exe) and the network server (dbsrv7.exe).
- Microsoft Windows NT 4.0 must be installed in its evaluated configuration as specified in its Trusted Facility Manual. (Follow links from the Trusted Products Evaluation Program (TPEP) website http://www.radium.ncsc.mil/tpep/ to access this document.)
- ASA must be installed on an NTFS partition on Windows NT.
- ASA must be installed in accordance with the Installation section of the *C2 Supplement.* This section specifies the options and capabilities that must be enabled or disabled for C2 operation.
- The C2 Update (also known as the C2 Patch) must be installed.
- ASA must be installed to run as a Windows NT Service.
- Communications between client applications and ASA must occur over Windows NT Named Pipes.
- Although the Windows NT machine can be networked, the ASA server cannot be accessed remotely. (ASA was only tested on a single machine and ASA cannot be accessed remotely over Named Pipes.)
- Users must use integrated login.
- Transaction logging is required.
- The trusted client interfaces include Interactive SQL, command line utilities, and the ESQL application programming interface.
- ASA must be maintained in accordance with the *C2 Supplement.*

11 Appendix B - Evaluated Product List Entry

SYBASE, INC.

ADAPTIVE SERVER ANYWHERE

VERSION 7.0.0

WITH C2 UPDATE

July 2000

TTAP EVALUATION FACILITY

Security Evaluation Laboratory, CygnaCom Solutions, an Entrust Technologies company, http://www.cygnacom.com/labs/sel.htm. Contact santosh.chokhani@entrust.com.

PRODUCT DESCRIPTION:

Sybase Inc.'s Adaptive Server Anywhere 7.0.0 with C2 Update (ASA) is a client-server relational database management system (DBMS). In the evaluated configuration, ASA executes on the Microsoft Windows NT 4.0 with Service Pack 6a and C2 Update operating system. ASA is Sybase's commercial DBMS for mobile computing. Both the ASA stand-alone engine and the ASA network server were evaluated. However, the ASA server cannot be accessed remotely in the evaluated configuration. The client and server were evaluated running on the same machine.

ASA provides an integrated login mechanism that maps the ASA user ID to the NT user ID. ASA's discretionary access control (DAC) mechanism controls access of users and groups to tables, views, procedures and functions. ASA provides an audit mechanism that is separate from the NT audit mechanism. ASA also has integrity features such as triggers, declarative and referential integrity, and transaction control. In addition, ASA includes a suite of tools for system administration.

The ASA relational database delivers rich database functionality, including full transaction processing, stored procedures, row-level locking, and Java support. ASA is designed to be self-tuning and yet maintain a small footprint. ASA symmetric multi-processor (SMP) support ensures top performance for greater numbers of users. A high-performance, self-tuning query optimizer determines the most effective way to access information and utilize additional processors, thereby improving performance and eliminating the need for expert tuning.

PRODUCT STATUS:

Adaptive Server Anywhere 7.0.0 with C2 Update has been available since March 2000. The evaluation contact for ASA at Sybase, Inc. is Mike Paola. Mike can be reached at (519) 883-6311 or by email at mpaola@sybase.com. ASA sales can be reached at 1-800-8-SYBASE or by email at dbtalk@sybase.com. Sybase 's web address is http://www.sybase.com.

SECURITY EVALUATION STATUS:

Sybase Inc.'s Adaptive Server Anywhere 7.0.0 with C2 Update hosted on Microsoft Windows NT 4.0 with Service Pack 6a and C2 Update was evaluated against the "DoD Trusted Computer System Evaluation Criteria," dated December 1985, as interpreted by the "Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria," dated April 1991, as a C2 class database management system.

The Security Evaluation Laboratory of CygnaCom Solutions has determined that the highest class at which ASA running on NT satisfies all specified requirements of the Criteria is class C2. For a complete description of how ASA running on NT satisfies each requirement of the TCSEC and TDI, see the Final Evaluation Report: Sybase, Inc., Adaptive Server Anywhere 7.0.0 with C2 Update.

EVIRONMENTAL STRENGTHS:

Adaptive Server Anywhere 7.0.0 with C2 Update provides an integrated login mechanism for user identification and authentication. This mechanism uses the Windows NT integrated login capabilities and grants database access to users by way of their NT user name and password. Integrated Login is the only permitted user authentication mechanism in the evaluated configuration.

ASA enforces a Discretionary Access Control (DAC) security policy on the subjects and objects under its control through the standard SQL GRANT and REVOKE mechanism. Database connections are the ASA subjects. ASA objects include tables, views, procedures, and functions. Object owners can grant and revoke object permissions to/from users and groups. Stored procedures, triggers, and views can be used to provide enhanced DAC capabilities by propagating the object access permissions of the procedure owner to the procedure user. This mechanism provides a means to specify and limit how users may interact with sensitive tables and databases. For example, a table owner can prohibit users from accessing the table directly, while allowing users to access a procedure that accesses that table in a restricted manner that the table owner specifies.

ASA provides the capability for auditing. An audit reduction tool allows audit records to be selected sequentially by user.

ASA provides two authorities: DBA (database administrator) and Resource. Users with DBA authority can perform trusted system administrative functions such as adding new users, granting authorities, and managing audit logs. By default, each newly created database includes the DBA user ID that is granted DBA authority. Resource authority permits users to create database objects.

12 Appendix C - SQL Interface Tables

Table 12.1 contains a list of the SQL statements along with the applicable interface, a brief description, and the required permission checks. Some statements can be used only in specific interfaces as shown in the column 'interface' in the following tables. These interfaces are as follows:

- ESQL: The statement is for use in Embedded SQL
- ISQL: The statement can be used only in Interactive SQL
- SP: The statement is for use in stored procedures, triggers, or batches
- TSQL: The statement is implemented for compatibility with Adaptive Server Enterprise

If the column is blank, the SQL statement can be used in all four interfaces.

Table 12.1 - SQL Statements

SQL Statement	Inter-face	Description of Functionality	Permissions
ALLOCATE DESCRIPTOR	ESQL	Allocates space for a descriptor area	None
ALTER DATABASE		Upgrade a Version 5 database to enable Java operations	DBA authority
ALTER DBSPACE		Modify characteristics of main db file or an extra dbspace or to preallocate space for a database or transaction log	DBA authority.
ALTER EVENT		Changes the definition of an event or its associated handler for automating predefined actions. Also, alters the definition of scheduled actions.	DBA Authority
ALTER PROCEDURE		To replace a procedure with a modified version	Owner of procedure or DBA
ALTER SERVER		Modify the attributes of a remote server	Resource authority
ALTER TABLE		Modify a table definition	Owner of table; or *ALTER* permission, Or DBA
ALTER TRIGGER		To replace a trigger definition with a modified version	Owner of the table on which the trigger is defined; ALTER permission and resource authority; or DBA
ALTER VIEW		To replace a view definition with a modified version	Owner of the view or DBA
ALTER WRITEFILE		Modify configuration of a writefile	Set using –gu option on server command line. Default is DBA, which is required in the evaluated configuration.
BACKUP		Back up a database and transaction log.	DBA authority.

SQL Statement	Inter-face	Description of Functionality	Permissions
BEGIN		Groups SQL statements. Used in body of a procedure or trigger.	None
BEGIN TRANSACTION		Begin a user-defined transaction	None.
CALL		To invoke a procedure	Owner of procedure, *EXECUTE* permission, or DBA
CASE		Select execution path	None
CHECKPOINT		Checkpoint a database	DBA authority for network server. None for stand-alone engine.
CLEAR	ISQL	Clear ISQL Data window.	None
CLOSE	ESQL, SP	Close a cursor	Cursor must have been previously opened.
COMMENT		Store comment in the system tables for a database object.	Owner of the database object being commented, or DBA
COMMIT		Make any changes to database permanent	None.
CONFIGURE	ISQL	Active ISQL configuration window.	None
CONNECT	ESQL, ISQL	To establish a connection to a database	None
CREATE COMPRESSED DATABASE		To create a compressed database from an existing database file or to expand a compressed database	Set using –gu option on the server command line. Default is DBA, which is required in evaluated configuration. Write permissions are required on the directories where files are created.
CREATE DATABASE		To create a database. The database is an operating system file	Set using –gu option on server command line. Default is DBA, which is required in the evaluated configuration. The account under which the server is running must have write permission on the directory where the files are created.
CREATE DBSPACE		To create a new database file. This file may be on a different device	DBA authority
CREATE DOMAIN		To create user-defined data type in the database	Resource authority
CREATE EVENT		To define an event and its associated handler for automating predefined actions. Also, to define scheduled actions.	DBA authority
CREATE EXISTING TABLE		Create a new proxy table representing an existing object on a remote server.	Resource Authority. To create a table for another user, must have DBA authority.
CREATE EXTERNLOGIN		To assign an alternate login name and password to be used when communicating with a remote server.	Only the login-name and the DBA account can add or modify an external login for login-name.
CREATE FUNCTION		To create a new function in the database	Resource authority. For external functions, DBA authority
CREATE INDEX		To create an index on a specified table	Owner of the table, REFERENCES permission, or DBA.
CREATE MESSAGE	TSQL	To add a user-defined message to the SYSUSERMESSAGES system table for use by PRINT and RAISEROR Calls	Resource authority

SQL Statement	Inter-face	Description of Functionality	Permissions
CREATE PROCEDURE		To create a procedure in the database	Resource authority. For external procedures, DBA authority
CREATE SCHEMA		Creates a collection of tables, views, and permissions for a database user	Resource authority
CREATE SERVER		To add a server to the SYSSERVERS system table.	Resource authority.
CREATE TABLE		Creates a table in a database.	Resource authority. DBA authority to create a table for another user.
CREATE TRIGGER		To create a new trigger in the database	Owner of table or ALTER permission and Resource authority or DBA authority
CREATE VARIABLE		To create a SQL variable	None
CREATE VIEW		To create a view on the database	Resource authority and SELECT permission on the tables in the view definition
CREATE WRITEFILE		To create a write file for a database	Set using –gu option on the server command line. Default is DBA, which is required in the evaluated configuration. The account under which the server is running must have write permissions on the directories where files are created.
DEALLOCATE DESCRIPTOR	ESQL	Frees memory associated with a SQL descriptor area	None
DECLARE		To declare a SQL variable with in a compound statement	None
DECLARE CURSOR	ESQL, SP	To declare a cursor	None
DECLARE LOCAL TEMPORARY TABLE		To declare a local temporary table	None
DELETE		To delete rows from the database	DELETE permission or DBA
DELETE (POSITIONED)	ESQL, SP	To delete the data at the current location of a cursor	DELETE permission on tables used in the cursor
DESCRIBE	ESQL	To get information about host variables	None
DISCONNECT	ESQL, ISQL	To drop a connection with the database	None
DROP CONNECTION		To drop a connection belonging to any user	DBA authority
DROP DATABASE		To remove a database	Set using –gu option on the server command line. Default is DBA, which is require in the evaluated configuration
DROP DATATYPE		To remove a datatype	Owner of datatype or DBA authority
DROP DBSPACE		To remove dbspace	DBA Authority
DROP DOMAIN		To remove a domain	Owner of domain or DBA authority
DROP EVENT		To remove an event	Owner of event or DBA authority
DROP EXTERNLOGIN		To drop an external login from ASA catalogs	Owner of login-name or have DBA authority.

SQL Statement	Inter-face	Description of Functionality	Permissions
DROP FUNCTION		Drop function from the database	Owner of function or DBA authority
DROP INDEX		Drop index from the database	Owner of table or REFERENCES permission on table or DBA authority
DROP MESSAGE		Delete message from database	Owner of message or DBA authority.
DROP OPTIMIZER STATISTICS		Erase all optimizer statistics	DBA authority
DROP PROCEDURE		Drop procedure from the database	Owner of procedure or DBA
DROP SERVER		Drop remote server from ASA catalog	DBA authority
DROP STATEMENT	ESQL	To free statement resources	Must have prepared the statement
DROP TABLE		Drop table from the database	Owner of the Table or DBA
DROP TRIGGER		Drop trigger from the database	Owner of the table or ALTER permission on the table or DBA authority
DROP VARIABLE		To eliminate SQL variable	None
DROP VIEW		To drop a view from the database	Owner of the View or DBA
EXECUTE	ESQL	To execute a prepared SQL statement	None. Permissions are checked on the statement being executed
EXECUTE	TSQL	To invoke a procedure, as an ASE compatible alternative to the CALL statement	Owner of the procedure, EXECUTE permission or DBA
EXECUTE IMMEDIATE	ESQL, SP	To enable dynamically constructed statements to be executed from within a procedure	None. The statement is executed with the permissions of the owner of the procedure, not with the permissions of the user who calls the procedure.
EXIT	ISQL	To leave Interactive SQL	None
EXPLAIN	ESQL	To retrieve a text specification of the optimization strategy used for a particular cursor	Must have opened the named cursor
FETCH	ESQL, SP	To reposition a cursor and then get data from it	Cursor must be opened and SELECT permission on the tables referenced in the declaration of the cursor
FOR		Repeat the execution of the statement list once for each row in a cursor	None
FORWARD TO		Sends native syntax to a remote server.	None
FROM		To specify the database tables or views involved in a SELECT or UPDATE statement	None
GET DATA	ESQL	To get string or binary data for one column on the current row of a cursor	Cursor must be opened and positioned on a row, using FETCH
GET DESCRIPTOR	ESQL	Retrieves information about a variable within a descriptor area or retrieves its value	None
GET OPTION	ESQL	To find current settings of an option	None
GOTO	TSQL	To branch to a labeled statement	None
		To grant special privileges to the user	DBA authority
GRANT CONNECT TO .. IDENTIFIED BY …		To create a new user	DBA authority or user to change own password

SQL Statement	Inter-face	Description of Functionality	Permissions
GRANT DBA		Grant DBA authority	DBA authority
GRANT RESOURCE		Grant Resource Authority	DBA authority
GRANT GROUP		Grant Group permission	DBA authority
GRANT MEMBERSHIP IN GROUP		Grant membership in group	DBA authority
GRANT INTEGRATED LOGIN TO		Grant integrated login permission to the user	DBA authority
GRANT [*TABLE PRIVILEGE*] ON ..		To grant permission on individual tables or views	Owner of the table/view or permission on the table/view WITH GRANT OPTION or DBA authority
GRANT EXECUTE ON		To grant permission to execute a procedure or function	Owner of the procedure/function or DBA authority
HELP	ISQL	To receive help in the interactive SQL environment	None
IF		To provide conditional execution of SQL statements	None
INCLUDE	ESQL	To include a file into a source program to be scanned by the SQL source language preprocessor	None
INPUT	ISQL	To import data into a database table from an external file or from the keyboard	Owner of table or INSERT permission or DBA authority
INSERT		To insert a single row or a selection of rows from elsewhere in the database into a table	Owner of table or INSERT permission or DBA authority
INSTALL		To make Java classes available for use within a database	DBA authority
LEAVE		Continue execution, by leaving a compound statement or LOOP	None
LOAD TABLE		To import data into a database table from an external ASCII-format file	Set using the -gl command line option. Default is DBA which is required in the evaluated configuration.
LOCK		to prevent other concurrent transactions from accessing or modifying a table.	To lock a table in SHARE mode, SELECT privileges are required. To lock a table in EXCLUSIVE mode, must be the table owner or have DBA authority.
LOOP		Repeat the execution of a statement list	None
MESSAGE		To display a message on the message window of the database server	None
OPEN	ESQL, SP	To open a previously declared cursor to access information from the database	SELECT permission on all tables in a SELECT statement or EXECUTE permission on the procedure in a CALL statement
OUTPUT	ISQL	To output the current query results to a file	None
PARAMETERS	ISQL	To specify parameters to a Interactive SQL command file	None
PREPARE	ESQL	To prepare a statement to be executed later or used for a cursor	None

SQL Statement	Inter-face	Description of Functionality	Permissions
PREPARE TO COMMIT		To check whether a COMMIT can be performed	None
PRINT	TSQL	To return a message to the client, or display a message in the message window of the database server.	None
PUT	ESQL, SP	To insert a row into the tables(s) specified by the cursor	Owner of Table, INSERT permission or DBA authority
RAISERROR	TSQL	To signal an error, and send a message to the client	None.
READ	ISQL	To read Interactive SQL statements from a file	None
READTEXT	TSQL	Reads text and image values, starting from a specified offset and reading a specified number of bytes	Owner of table or SELECT permission or DBA authority.
RELEASE SAVEPOINT		Release a savepoint within the current transaction	There must have been a corresponding SAVEPOINT within the current transaction.
REMOVE		To remove a class, a package or a jar file from a database	DBA authority
RESIGNAL		Resignal an exception condition	None
RESTORE		To restore a backed up database from an archive.	Must be connected to the utility database.
RESUME		To resume a procedure following a query	The cursor must have been previously opened
RETURN		To exit from a function or procedure unconditionally	None
REVOKE		To remove permissions for specified user(s)	Granter of the permissions that are being revoked or DBA authority
ROLLBACK		To undo any changes made since last COMMIT or ROLLBACK	None
ROLLBACK TO SAVEPOINT		To cancel any changes made since a SAVEPOINT	None
ROLLBACK TRIGGER		To undo any changes made in a trigger	None
SAVEPOINT		To establish a savepoint within the current transaction	None
SELECT		To retrieve information from the database	Owner of table/view or SELECT permission or DBA
SET	TSQL	To set database options in an ASE compatible manner	None
SET		To assign a value to a SQL variable	None
SET CONNECTION	ISQL, ESQL	To change the active database connection	None
SET DESCRIPTOR	ESQL	Describes the variables in a SQL descriptor area, and places data into the descriptor area	None
SET OPTION		Change database option	None for own options. DBA to set options for another user or PUBLIC
SET OPTION	ISQL	To change Interactive SQL options	None
SET SQLCA	ESQL	To tell the SQL preprocessor to use a SQLCA other than the default global sqlca	None

SQL Statement	Inter-face	Description of Functionality	Permissions
SETUSER		To allow a database administrator to impersonate another user, and to enable connection pooling	DBA authority.
SIGNAL		Signal an exception condition	None
START DATABASE		Start a database on the specified database server	Specified by -gd switch. DBA authority in evaluated configuration.
START ENGINE	ISQL	Start a database server	None
START JAVA		Start Java VM	None
START LOGGING	ISQL	Start logging executed SQL statements to a log file.	None
STOP DATABASE		Stop a database on the specified database server	Specified by -gd switch. DBA authority in evaluated configuration.
STOP ENGINE		Stop a database server	Specified by -gd switch. DBA authority in evaluated configuration.
STOP JAVA VM		Stop Java VM	None
SYSTEM	ISQL	To execute an operating system command from within Interactive SQL	None
TRIGGER EVENT		To trigger a named event	DBA authority
TRUNCATE TABLE		To delete all rows from a table, without deleting the table definition	Owner of the table or ALTER permission or DBA
UNION		To combine the results of two or more select statements	Must have SELECT permission for each of the component SELECT statements
UNLOAD		To export data from a database into an external ASCII-format file	Set using the -gl command line option. Default is DBA which is required in the evaluated configuration.
UNLOAD TABLE		To export data from a database table into an external ASCII format file	Set using the -gl command line option. Default is DBA which is required in the evaluated configuration.
UPDATE		To modify existing rows in database tables	Table owner or UPDATE permission on the columns being modified or DBA
UPDATE (POSITIONED)		To modify the data at the current location of a cursor	Table owner or UPDATE permission on the columns being modified or DBA
VALIDATE INDEX		Checks that rows listed in an index actually exist in a table.	Owner of table or DBA authority or REMOTE DBA authority
VALIDATE TABLE		To validate a table in the database	Owner of the table, DBA authority, or REMOTE DBA authority
WHENEVER	ESQL	To specify error handling in an Embedded SQL program	None
WHILE	TSQL	To provide repeated execution of a statement or compound statement	None
WRITETEXT	TSQL	Permits non-logged, interactive updating of an existing text or image column	None

1

13 Appendix D - System Stored Procedures

Table 13.1 documents the team's analysis of stored procedures. We classified the stored procedures using the following codes in the Comments column:

- 1 = Does nothing but call sp_tsql_feature_not_supported.
- 2 = Calls sp_checkperms with input "DBA" as a first step.
- 3 = Calls sp_checkperms with input "RESOURCES" as a first step.
- 4 = Calls two dll files from Windows
- 5 = Calls another stored procedure
- 6 = Uses "internal name" plus variable syntax

The Comments column contains a short synopsis of the body when no code applies.

Table 13.1 - System Stored Procedures

Proc_id	Creator	Grantee	Procedure Name	Comments	Parameters
1	dbo	Public(2)	sp_tsql_feature_not_supported	declares an exception for sqlstate value "0AW02"	none
2	dbo	Public(2)	sp_checkperms	checks for DBA or Resource as input. If input is not either, then signals permission denied	in: required_auth
3	dbo	Public(2)	sp_addalias	1	in: @login_name, in: @name
4	dbo	Public(2)	sp_addauditrecord	1	in: @text, in: @db_name, in: @obj_name, in: @owner_name, in: @dbid
5	dbo	Public(2)	sp_addgroup	2	in: @grpname
6	dbo	Public(2)	sp_addlanguage	1	in: @language, in: @alias in: @months, in: @shortmons in: @days, in: @datefmt, in: @datefirst
7	dbo	Public(2)	sp_addlogin	2	in: @login_name, in: @passwd in: @defaultdb, in: @deflanguage in: @fullname
8	dbo	Public(2)	sp_addmessage	3	in: @message_num in: @message_text, in:@language

Proc_id	Creator	Grantee	Procedure Name	Comments	Parameters
9	dbo	Public(2)	sp_addremotelogin	1	in: @remoteserver in: @login_name in: @remotename
10	dbo	Public(2)	sp_addsegment	1	in: @segname, in: @dbname in: @device_name
11	dbo	Public(2)	sp_addserver	1	in: @server, in: @local_null in: @network_name
12	dbo	Public(2)	sp_addthreshold	1	in: @database, in: @segment in: @free_pages, in: @procedure
13	dbo	Public(2)	sp_addtype	3	in: @typename, in: @phystype in: @ident_null
14	dbo	Public(2)	sp_addumpdevice	1	in: @tape_disk, in: @device_name in: @phys_name, in: @size
15	dbo	Public(2)	sp_adduser	2	in: @login_name, in: @name_in_db in: @grpname
16	dbo	Public(2)	sp_auditdatabase	1	in: @dbname, in: @audittype, in: @event_types
17	dbo	Public(2)	sp_auditlogin	1	in: @login_name, in: @table_view, in: @audittype, in: @cmdtext, in: @on_off
18	dbo	Public(2)	sp_auditobject	1	in: @objname, in: @dbname in: @audittype, in: @event_types
19	dbo	Public(2)	sp_auditoption	1	in: @auditopt, in: @optval
20	dbo	Public(2)	sp_auditsproc	1	in: @sp_name, in: @dbname in: @audittype
21	dbo	Public(2)	sp_bindefault	1	in: @defaultname, in: @objectname in: @futureonly
22	dbo	Public(2)	sp_bindmsg	1	in: @constraint_name in: @message_num
23	dbo	Public(2)	sp_bindrule	1	in: @rulename, in: @objectname in: @futureonly
24	dbo	Public(2)	sp_changedbowner	1	in: @login_name in: @true_kw
25	dbo	Public(2)	sp_changegroup	2	in: @grpname in: @name_in_db
26	dbo	Public(2)	sp_checknames	1	none
27	dbo	Public(2)	sp_checkreswords	1	in: @user_name
28	dbo	Public(2)	sp_clearstats	1	in: @user_name

Proc_id	Creator	Grantee	Procedure Name	Comments	Parameters
29	dbo	Public(2)	sp_commonkey	1	in: @tabaname, in: @tabbname, in: @col1a, in: @col1b, in: @col2a, in: @col2b, in: @col3a, in: @col3b in: @col4a, in: @col4b, in: @col5a, in: @col5b, in: @col6a, in: @col6b in: @col7a, in: @col7b
30	dbo	Public(2)	sp_configure	1	in: @config_name, in: @config_value
31	dbo	Public(2)	sp_cursorinfo	1	in: @cursor_level, in: @cursor_value
32	dbo	Public(2)	sp_dboption	if nulls, then 1 otherwise looks at “Allow nulls by default”	in: @dbname, in: @optname, in: @true_false
33	dbo	Public(2)	sp_dbremap	1	in: @database_name
34	dbo	Public(2)	sp_depends	1	in: @objname
35	dbo	Public(2)	sp_diskdefault	1	in: @logical_name, in: @dfit_on_off
36	dbo	Public(2)	sp_displaylogin	1	in: @login_name
37	dbo	Public(2)	sp_dropalias	1	in: @login_name
38	dbo	Public(2)	sp_dropdevice	1	in: @device_name
39	dbo	Public(2)	sp_dropgroup	2	in: @grpname
40	dbo	Public(2)	sp_dropkey	1	in: @keytype, in: @tabaname in: @tabbname
41	dbo	Public(2)	sp_droplanguage	1	in: @language, in: @dropmessages
42	dbo	Public(2)	sp_droplogin	2	in: @login_name
43	dbo	Public(2)	sp_dropmessage	3	in: @message_number in: @language
44	dbo	Public(2)	sp_dropremotelogin	1	in: @remoteserver, in: @login_name, in: @remotename
45	dbo	Public(2)	sp_dropsegment	1	in: @segname, in: @dbname, in: @device_name
46	dbo	Public(2)	sp_dropserver	1	in: @server, in: @droplogins
47	dbo	Public(2)	sp_dropthreshold	1	in: @database, in: @segment_name in: @free_pages
48	dbo	Public(2)	sp_droptype	3	in: @typename
49	dbo	Public(2)	sp_dropuser	2	in: @name_in_db
50	dbo	Public(2)	sp_estspace	1	in: @table_name, in: @no_of_rows in: @fill_factor, in: @cols_to_max in: @textbin_len, in: @iosec
51	dbo	Public(2)	sp_extendsegment	1	in: @segname, in: @dbname, in: @device_name

Proc_id	Creator	Grantee	Procedure Name	Comments	Parameters
52	dbo	Public(2)	sp_foreignkey	1	in: @tabname, in: @pktabname, in: @col1, in: @col2, in: @col3, in: @col4, in: @col5, in: @col6, in: @col7, in: @col8
53	dbo	Public(2)	sp_getmessage	gets msg desc from db table	in: @message_num, out: @msg_var, in: @language
54	dbo	Public(2)	sp_help	1	in: @objname
55	dbo	Public(2)	sp_helpconstraint	1	in: @tabname, in: @detail
56	dbo	Public(2)	sp_helpdb	1	in: @dbname
57	dbo	Public(2)	sp_helpdevice	1	in: @device_name
58	dbo	Public(2)	sp_helpgroup	1	in: @grpname
59	dbo	Public(2)	sp_helpindex	1	in: @tabname
60	dbo	Public(2)	sp_helpjoins	1	in: @lefttab in: @righttab
61	dbo	Public(2)	sp_helpkey	1	in: objname
62	dbo	Public(2)	sp_helplanguage	1	in: @language
63	dbo	Public(2)	sp_helplog	1	none
64	dbo	Public(2)	sp_helpremotelogin	1	in: @remoteserver in: @remotename
65	dbo	Public(2)	sp_helpprotect	1	in: @name, in: @name_in_db, in: @grant_kw
66	dbo	Public(2)	sp_helpsegment	1	in: @segname
67	dbo	Public(2)	sp_helpserver	1	in: @server
68	dbo	Public(2)	sp_helpsort	1	none
69	dbo	Public(2)	sp_helptext	find text using objid	in: @objname
70	dbo	Public(2)	sp_helpthreshold	1	in: @segment_name
71	dbo	Public(2)	sp_helpuser	1	in: @name_in_db
72	dbo	Public(2)	sp_indsuspect	1	in: @table_name
73	dbo	Public(2)	sp_lock	1	in: @spid1 in: @spid2
74	dbo	Public(2)	sp_locklogin	1	in: @login_name in: @lock_unlock
75	dbo	Public(2)	sp_logdevice	1	in: @dbname in: @device_name
76	dbo	Public(2)	sp_modifylogin	1	in: @login_name in: @optname in: @value
77	dbo	Public(2)	sp_modifythreshold	1	in: @database, in: @segment, in: @free_pages, in: @new_procedure in: @new_free_pages in: @new_segment

Proc_id	Creator	Grantee	Procedure Name	Comments	Parameters
78	dbo	Public(2)	sp_monitor	1	none
79	dbo	Public(2)	sp_password	2	in: @caller_pswd, in: @new_pswd in: @login_name
80	dbo	Public(2)	sp_placeobject	1	in: @segname, in: @objname
81	dbo	Public(2)	sp_primarykey	1	in: @tabname, in: @col1, in: @col2, in: @col3, in: @col4, in: @col5, in: @col6, in: @col7, in: @col8
82	dbo	Public(2)	sp_procxmode	1	in: @proc_name, in: @trans_mode
83	dbo	Public(2)	sp_recompile	1	in: @tabname
84	dbo	Public(2)	sp_remap	1	in: @object_name
85	dbo	Public(2)	sp_remoteoption	1	in: @remoteserver in: @login_name, in: @remote_name in: @opt_name, in: @true_false
86	dbo	Public(2)	sp_rename	1	in: @objname, in: @newname
87	dbo	Public(2)	sp_renamedb	1	in: @dbname, in: @newname
88	dbo	Public(2)	sp_reportstats	1	in: @user_name
89	dbo	Public(2)	sp_role	1	in: @grant_revoke in: @role_type, in: @login_name
90	dbo	Public(2)	sp_serveroption	1	in: @server, in: @optname in: @true_false
91	dbo	Public(2)	sp_setlangalias	1	in: @language, in: @alias
92	dbo	Public(2)	sp_spaceused	1	in: @table_name
93	dbo	Public(2)	sp_syntax	1	in: @cmd_or_frag, in: @module_name, in: @language
94	dbo	Public(2)	sp_unbindefault	1	in: @objname, in: @futureonly
95	dbo	Public(2)	sp_unbindmsg	1	in: @constraint_name
96	dbo	Public(2)	sp_unbindrule	1	in: @objname, in: @futureonly
97	dbo	Public(2)	sp_volchanged	1	in: @session_id, in: @device_name in: @action, in: @filename, in: @volume_name
98	dbo	Public(2)	sp_who	1	in: @login_name
99	dbo	Public(2)	sp_column_privileges	1	in: @table_name, in: @table_owner in: @table_qualifier, in: @column_name
100	dbo	Public(2)	sp_columns	produces a result with info from system tables	in: @table_name in: @table_owner in: @table_qualifier in: @column_name
101	dbo	Public(2)	sp_databases	1	none
102	dbo	Public(2)	sp_datatype_info	1	in: @data_type

Proc_id	Creator	Grantee	Procedure Name	Comments	Parameters
103	dbo	Public(2)	sp_fkeys	produces a result with info from system tables	in: @pktable_name, in: @pktable_woner in: @pktable_qualifier, in: @fktable_name, in: @fktable_owner in: @fktable_qualifier
104	dbo	Public(2)	sp_pkeys	produces a result with info from system tables	in: @table_name, in: @table_owner in: @table_qualifier
105	dbo	Public(2)	sp_serverinfo	gets info from sys table	@request
106	dbo	Public(2)	sp_server_info	1	
107	dbo	Public(2)	sp_special_columns	produces result with info from system tables	in: @table_name, in: @table_owner in: @table_qualifier, in: @col_type
108	dbo	Public(2)	sp_sproc_columns	produces result with info from system tables	in: @sp_name, in: @sp_owner in: @sp_qualifier, in: @column_name
109	dbo	Public(2)	sp_statistics	produces result with info from system tables	in: @table_name, in: @table_owner, in: @table_qualifier, in: @index_name, in: @is_unique
110	dbo	Public(2)	sp_stored_procedures	produces result with info from system tables	in: @sp_name, in: @sp_owner, in: @sp_qualifier
111	dbo	Public(2)	sp_table_privileges	1	in: @table_name, in: @table_owner, in: @table_qualifier
112	dbo	Public(2)	sp_tables	produces result with info from system tables	in: @table_name, in: @table_owner in: @table_qualifier, in: @table_type
113	dbo	Public(2)	sp_reset_tsql_environment	sets temporary options for DateTime formats, Date Order YMD, Escape Char on, Close on endtrans on, etc.	none
114	dbo	Public(2)	sp_tsql_environment	sets same temporary options as 113	none
115	dbo	Public(2)	sp_login_environment	5: if TDS, then calls 114	none
116	dbo	none	col_length	returns an integer from info from a system table	in: @object_name in: @column_name
117	dbo	none	col_name	gets info from syscolumns	in: @object_id, in: @column_id in: @database_id
118	dbo	none	index_col	gets info from sys tables	in: @object_name, in: @index_id in: @key_#, in: @user_id
119	dbo	none	object_id	returns an integer from info selected from dbo and SYS tables	in: @object_name
120	dbo	none	object_name	gets info from sysobjects	in: @object_id in: @database_id
121	dbo	none	proc_role	only returns 0	in: @role_type
122	dbo	none	show_role	only returns null	none
123	dbo	none	xp_startmail	4	in: mail_user, in: mail_password

Proc_id	Creator	Grantee	Procedure Name	Comments	Parameters
124	dbo	none	xp_stopmail	4	none
125	dbo	none	xp_sendmail	4	in: recipient, in: subject, in: cc_recipient, in: bcc_recipient, in: query, in: “message”, in: attachment, in: attach_result, in: echo_error, in: include_file, in: no_column_header, in: no_output, in: width, in: separator, in: dbuser
126	dbo	none	xp_cmdshell	4	in: command, in: redir_output
127	dbo	Public(2)	xp_sprintf	4, for Windows 95, Windows NT, Netware, or Windows 3x.x	out: output_buffer, in: format, in: parm1, in: parm2, in: parm3, in: parm4, in: parm5, in: parm6, in: parm7, in: parm8, in: parm9, in: parm10, in: parm11, in: parm12, in:parm13, in:parm14
128	dbo	Public(2)	xp_scanf	4, for Windows 95, Windows NT, Netware, or Windows 3x.x	in: input_buffer, in: format out: parm1, out: parm2, out: parm3, out: parm4, out: parm5, out: parm6 out: parm7, out: parm8, out: parm9, out: parm10, out: parm11, out: parm12, out: parm13, out: parm14
129	dbo	none	xp_msver	gets information on Product Names, Versions, Trademarks, and Copywrites	in: the_option
130	dbo	unknown	xp_write_file	4; requires DBA authority	in: filename, in: file_contents, returns int
131	dbo	unknown	xp_read_file	4; requires DBA authority	Filename; returns long binary
132	dbo	Public(2)	sa_db_info	produces a result using info from SYSTEM	in: dbidparm
133	dbo	Public(2)	sa_conn_info	produces a result using info from SYSTEM	in: connidparm
134	dbo	Public(2)	sa_eng_properties	produces a result using info from SYSTEM	none
135	dbo	Public(2)	sa_db_properties	produces a result using info from SYSTEM	in: dbidparm
136	dbo	none	internal_sa_db_properties	produces a result from values in next_connection and connection_properties	in: connidparm
137	dbo	Public(2)	sa_conn_properties	5	in connidparm
138	dbo	none	sa_conn_properties_by_name	5	in: connidparm
139	dbo	none	sa_conn_properties_by_conn	5	in: prop_name
140	dbo	none	sa_validate	gets info from sys tables	in: tbl_name, in: owner_name in: check_type
141	dbo	none	sa_internal_table_page_usage	6	none
142	dbo	none	sa_table_page_usage	5	none

Proc_id	Creator	Grantee	Procedure Name	Comments	Parameters
143	dbo	unknown	sa_internal_index_levels	6	in: tbl_name, in: owner_name
144	dbo	unknown	sa_index_levels	assists in performance tuning by reporting the number of levels in an index. DBA authority required	in: tbl_name, in: owner_name
145	sys(0)	none	sa_setremoteuser	updates SYSREMOTEUSER	p_user_id, p_log_sent, p_confirm_sent, p_send_count, p_resend_count, p_log_received, p_confirm_received, p_receive_count, p_rereceive_count
146	sys (0)	none	sa_setsubscription	update SYSSUBSCRIPTION	p_publication_id, p_user_id, p_subscribe_by, p_created, p_started
147	dbo	none	java_debug_version	6	out: version
148	dbo	SA_DEBUG(5)	java_debug_connect	6	in: user_to_debug
149	dbo	SA_DEBUG(5)	java_debug_disconnect	6	none
150	dbo	SA_DEBUG(5)	java_debug_get_existing_vms	6	out: buffer
151	dbo	SA_DEBUG(5)	java_debug_free_existing_vms	6	none
152	dbo	SA_DEBUG(5)	java_debug_wait_for_debuggable_vm	6	out: buffer
153	dbo	SA_DEBUG(5)	java_debug_get_vm_name	6	in: vm_handle, out: vm_name
154	dbo	SA_DEBUG(5)	java_debug_release_vm	6	in: vm_handle
155	dbo	SA_DEBUG(5)	java_debug_attach_to_vm	6	in: vm_name, out: debugger
156	dbo	SA_DEBUG(5)	java_debug_detach_from_vm	6	in: debugger
157	dbo	SA_DEBUG(5)	java_debug_request	6	in: debugger, in: request out: out_request
158	dbo	none	sa_proc_debug_version	6	out_version
159	dbo	SA_DEBUG(5)	sa_proc_debug_connect	6	in: user_to_debug
160	dbo	SA_DEBUG(5)	sa_proc_debug_disconnect	6	none
161	dbo	SA_DEBUG(5)	sa_proc_debug_wait_for_connection	6	out: buffer
162	dbo	SA_DEBUG(5)	sa_proc_debug_get_connection_name	6	in: conn_handle out: conn_name

Proc_id	Creator	Grantee	Procedure Name	Comments	Parameters
163	dbo	SA_DEBUG(5)	sa_proc_debug_release_connection	6	in: connection_handle
164	dbo	SA_DEBUG(5)	sa_proc_debug_attach_to_connection	6	in: connection_handle out: debugger
165	dbo	SA_DEBUG(5)	sa_proc_debug_detach_from_connection	6	in: debugger
166	dbo	SA_DEBUG(5)	sa_proc_debug_request	6	in: debugger
167	dbo	none	sa_flush_cache	6	none
168	dbo	none	sa_server_option	6	in: opt, in: val
169	dbo	none	sa_exec_script	6	in: filename
170	dbo	none	sa_jdk_version	produces a result with info from system table	none
171	dbo	none	sa_internal_read_backup_history	6	none
172	dbo	none	sa_read_backup_history	5; calls itself and selects from BackupOps	none
173	dbo	Public(2)	sp_remote_columns	5;calls itself using the “at anyserver…” syntax from Sybase email: In this case, [and for the other sps where anyserver is noted] "anyserver" is a dummy server name that ensures that the local RDA code handles the procedure call, rather than the regular stored procedure code.	in: @server_name, in: @table_name, in: @table_owner in: @table_qualifier
174	dbo	Public(2)	sp_remote_tables	5;calls itself using the “at anyserver…” syntax	in: @server_name, in: @table_name, in: @table_owner, in: @table_qualifier
175	dbo	Public(2)	sp_remote_pcols	5;calls itself using the “at anyserver….” syntax	in: @server_name, in: @sp_name in: @sp_owner, in: @sp_qualifier
176	dbo	Public(2)	sp_remote_procedures	5;calls itself using the “at anyserver….” syntax	in: @server_name, in: @sp_name in: @ssp_owner, in: @sp_qualifier
177	dbo	Public(2)	sp_servercaps	gets info from system tables	in: @sname
178	dbo	Public(2)	sa_forward_to	uses the ‘at anyserver…sp_forward_to’ syntax. No sp_forward_to is defined in sysprocedures table, but may be at another server not in the evaluated configuration. The user uses the "FORWARD TO" SQL statement to tell ASA to forward a string onto a remote server (since it may contain syntax that ASA doesn't support), and the engine uses this procedure to implement the forward to call.	in: @server_name
179	dbo	Public(2)	sa_end_forward_to	6	none
180	dbo	none	sa_audit_string	6	in: string
181	dbo	none	sa_internal_locks	6	in: connection, in: table_name in: max_locks
182	dbo	none	sa_locks	5; also gets info from sa_locks_table	in: connection, in: table_name in: max_locks

Proc_id	Creator	Grantee	Procedure Name	Comments	Parameters
183	dbo	none	sa_sync	6	in: user_id, in: operation in: value
184	dbo	none	sa_app_registration_unlock	6	none
185	dbo	none	sa_app_register	6	out: cookie, in: app_name in: conn_label, in: exclusive
186	dbo	none	sa_app_deregister	6	in: cookie
187	dbo	unknown	sa_app_get_status	6; Gets the status of all connections registered to the given app. Only used by dbsync and dbremote. Requires DBA authority.	in: app_name out: all_on out: all_off
188	dbo	unknown	sa_app_set_infoStr	6; Changes the per-app information associated with the given app. Only used by dbsync and dbremote. Requires DBA authority	in: app_info_str
189	dbo	unknown	sa_app_get_infoStr	6; Retrieves the app's info. Only used by dbsync and dbremote. Requires DBA authority.	in: app_name, out: app_info_str
190	dbo	none	sa_conn_register	6	in: cookie
191	dbo	none	sa_conn_deregister	6	in: cookie, in: conn_label
192	dbo	unknown	sa_conn_set_status	6; Sets the status of a connection. Only used by dbsync and dbremote. Requires DBA authority.	in: conn_status in: log_status_chg
193	dbo	none	sa_check_commit	6	out: tname out: keyname
194	dbo	Public(2)	sa_event_schedules	produces a result from a cursor that reads through sys.sysschedule and sys.dummy	in: evt_id
195	rs_systabgroup(4)	none	rs_get_lastcommit	does select from rs_lastcommit table	none
196	rs_systabgroup(4)	none	rs_update_lastcommit	does insert into rs_lastcommit table	in: @origin, in: @origin_qid in: @secondary_qid in: @origin_time
197	rs_systabgroup(4)	none	rs_marker	sets rs_api to rs_ap	inout: @rs_api
198	rs_systabgroup(4)	none	rs_initialize_threads	deletes and then inserts new row with value to 0 for rs_id in rs_threads table	in: @rs_id
199	rs_systabgroup(4)	none	rs_update_threads	updates seq value for rs_id in rs_threads table	in: @rs_id in: @rs_seq
200	dbo	none	ul_delete_project	deletes entry from ul_file based on input variable	in: @project
201	dbo	none	ul_add_project	inserts entry into ul_file after calculating a file id	in: @project
202	dbo	none	ul_delete_statement	5;calls sp_create_project and updates ul_statement	in: @project in: @name
203	dbo	none	ul_check_syntax	checks an input text string against a select	in: @statement
204	dbo	none	ul_add_statement	5;calls sp_create_project and sp_delete_project and updates ul_statement	in: @project, in: @name in: @statement

Proc_id	Creator	Grantee	Procedure Name	Comments	Parameters
205	dbo	none	ml_add_table_script	updates dbo.ul_script_version and dbo.ul_table	in: @version, in: @name in: @event, in: @script
206	dbo	none	ml_add_connection_script	updates dbo.ul_script_version, dbo.ul_script and dbo.ul_connection_script	in: @version, in: @event in: @script
208	dbo	Public(2)	sp_jconnect_trimit	returns a string up to 255 char based on db_property FileVersion	in: @iString
209	dbo	Public(2)	sp_jdbc_datatype_info	selects information from dbo_spt_jdatatype_info	none
210	dbo	Public(2)	sp_jdbc_function_escapes	implements dbo.jdbc_function_escapes	none
211	dbo	Public(2)	sp_jdbc_escapeliteralforlike	returns a new string created from the specified string with all the special characters escaped. Basically it puts a '\' character in front of every '%', '_', '\', '[', and ']' character. Used by other JDBC stored procedures so that they can use table and procedure names that contain these characters in queries against system tables.	output: @pString
212	dbo	Public(2)	sp_jdbc_tables	selects information from system tables	@table_name, @table_owner @table_qualifier, @table_type
213	dbo	Public(2)	sp_jdbc_columns	5;calls sp_jconnect_trimit with various parameters from input	@table_name, @table_owner @table_qualifier, @column_name
214	dbo	Public(2)	sp_mda	sets input variables to calculated numbers	@requesttype, @requestversion @clientversion = 0
215	dbo	Public(2)	sp_jdbc_fkeys	updates table dbo.jdbc_helpkeys	@pktable_name, @pktable_owner @pktable_qualifier, @fktable_name @fktable_owner, @fktable_qualifier
216	dbo	Public(2)	sp_jdbc_exportkey	5;executes sp_jdbc_escapeliteralforlike and gets information from system tables	@table_qualifier, @table_owner, @table_name
217	dbo	Public(2)	sp_jdbc_importkey	5; calls sp_jdbc_escapeliteralforlike and gets information from system tables	@table_qualifier, @table_owner, @table_name
218	dbo	Public(2)	sp_jdbc_getcrossreferences	5; also, selects information from dbo system and jdbc tables	@pktable_qualifier, @pktable_owner, @pktable_name @fktable_qualifier, @fktable_owner @fktable_name
220	dbo	Public(2)	sp_jdbc_convert_datatype	calculates new datatype from old datetype using selects from dbo.spt_jdbc_conversion	@source @destination
221	dbo	Public(2)	sp_jdbc_getprocedurecolumns	deletes from or inserts into table dbo.jdbc_procedurecolumns	@sp_qualifier, @sp_owner, @sp_name, @sp_column_name
222	dbo	Public(2)	sp_jdbc_primarykey	gets information about tables owned by input parameter from sysobjects	@table_qualifier, @table_owner
223	dbo	Public(2)	sp_jdbc_stored_procedures	5; calls sp_jconnect_trimit with various parameters selected from system tables	@sp_qualifier, @sp_owner @sp_name
224	dbo	Public(2)	sp_jdbc_gettableprivileges	produces a result by selecting from system tables	@table_qualifier, @table_owner @table_name

Proc_id	Creator	Grantee	Procedure Name	Comments	Parameters
225	dbo	Public(2)	sp_jdbc_getcolumnprivileges	selects information from system tables	@table_qualifier, @table_owner, @table_name, @column_name
226	dbo	Public(2)	sp_jdbc_getbestrowidentifier	selects information from system tables	@table_qualifier, @table_owner @table_name, @scope @nullable
227	dbo	Public(2)	sp_jdbc_getversioncolumns	selects information from system tables	@table_qualifier, @table_owner @table_name
228	dbo	Public(2)	sp_jdbc_getindexinfo	selects information from system tables	@table_qualifier, @table_owner, @table_name, @approximate
229	dbo	Public(2)	sp_sql_type_name	calculates datatype based on numeric value or selects information from sysusertype	@datatype @usrtype
230	dbo	Public(2)	sp_default_charset	selects information from dbo.spt_collation_map and sysinfo	none
231	dbo	Public(2)	sp_jdbc_getudts	uses a select to build something where 1=2. from Sybase email: it's not supported in ASA. It returns an ermpty result set.	@table_qualifier, @table_owner @type_name_pattern, @types
232	dba (1)	none	sp_retrieve_contacts	produces a report based on a select on contact table	none
233	dba (1)	none	sp_product_info	produces a result based on a select on prod_id from product table	inout: prod_id
234	dba (1)	none	sp_customer_list	produces a result based on select on customer table	none
235	dba (1)	none	sp_contacts	inserts or deletes or modifies an entry in the contact table	in: action, in: contact_id in: contact_old_id, in: contact_last_name, in: contact_first_name, in: contact_title, in: contact_street, in: contact_city, in: contact_state in: contact_zip, in: contact_phone in: contact_fax
236	dba (1)	none	sp_sales_order_items	produces a result based on select of info from the sales_order_items table based on order id	in: ord_id, in: product
237	dba (1)	none	sp_sales_order	produces a result based on select from sales_order_items and sales_order tables.	in: customer_id, in: product_id
238	dba (1)	none	sp_customer_products	produces a result based on a join of product, sales_order_items, and sales_order.	inout: customer_id

14 Appendix E - DBLIB Interface Tables

Table 14.1 documents the DBLIB interfaces including a brief description and the required permission checks.

Table 14.1 - DBLIB Functions

Function	Description	Permissions
Interface Initialization Functions		
db_init	Initializes the database interface library.	None.
db_fini	Frees resources used by the database interface or DLL	None.
Connection and Server Management Functions		
db_string_connect	Provides extra functionality beyond the Embedded SQL Connect command. Starts server and database if not running.	None.
db_string_disconnect	Disconnects from database	None.
db_start_engine	Starts a server if not already running.	None.
db_start_database	Starts a database on an existing server if the database is not already running.	Set on server command line with –gd switch. Set to DBA in evaluated configuration.
db_stop_database	Stops the identified database. By default does not stop database is there are still connections.	Set on server command line with –gk switch. Set to DBA in evaluated configuration.
db_stop_engine	Terminates execution of the database server.	Set on server command line with –gk switch. Set to DBA in evaluated configuration.
db_find_engine	Returns status information about the database server.	None.
SQLDA Management Functions		
alloc_sqlda_noind	Allocates a SQLDA with descriptors. Does not allocate space for indicator variables.	None.
alloc_sqlda	Allocates a SQLDA with descriptors. Allocates space for indicator variables.	None.

Function	Description	Permissions
fill_sqlda	Allocates space for each variable described in each descriptor of SQLDA.	None.
sqlda_string_length	Returns the length of the C string required to hold the variable.	None.
sqlda_storage	Returns the amount of storage required to store any value for the variable.	None.
fill_s_sqlda	Allocates space to hold string representation for each variable described in each descriptor of SQLDA	None.
free_filled_sqlda	Free the memory allocated to each sqldata pointer.	None.
free_sqlda_noind	Free space that was allocated to this sqlda. Indicator variable pointers are ignored.	None.
free_sqlda	Free space allocated to this sqlda.	None.
Backup Functions		
db_backup	Performs backup functions.	User ID with DBA authority.
db_delete_file	Used to delete old transaction log.	User ID with DBA authority
Asynchronous Functions		
db_cancel_request	Cancels the currently active database server request.	None.
db_is_working	Returns "1" if application has request in progress, "0" otherwise.	None.
Other Functions		
sql_needs_quotes	Returns whether or not string needs double quotes when used as SQL identifier.	None.
sqlerror_message	Returns a pointer to a string that contains an error message.	None.

1

15 Appendix F - Acronyms

ADP	Automated Data Processing
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASA	Adaptive Server Anywhere
CD	Compact Disk
CMD	Command
DAC	Discretionary Access Control
DB	Database
DBA	Database Administrator
DBLIB	Database Library
DBMS	Database Management System
DDL	Data Definition Language
DLL	Dynamic Link Library
DML	Data Manipulation Language
DU	Data Unit
EPL	Evaluated Products List
ESQL	Embedded SQL
FER	Final Evaluation Report
ID	Identification
I/O	Input/Output
IP	Internet Protocol
IPAR	Initial Product Assessment Report
ISQL	Interactive SQL
JAR	Java ARchive
JDBC	Java Database Connectivity
JDK	Java Development Kit
JNI	Java Native Interface
JVM	Java Virtual Machine
LRU	Least Recently Used
MAPI	Mail Application Program Interface
MSDN	Microsoft Data Network
NSA	National Security Agency
NT	New Technology
NTFS	NT File System
ODBC	Open Database Connectivity
OS	Operating System
PAT	Process Action Team
PGWG	Process Action Team Guidance Working Group
RC	Read Committed
RR	Repeatable Read
RU	Read Uncommitted
SEQ	Sequence
SFUG	Security Features Users Guide
SMP	Symmetric Multi-Processor
SP	Stored Procedure
SQL	Structured Query Language
SSPI	Security Support Provider Interface
TCB	Trusted Computing Base
TCP	Transmission Control Protocol
TCSEC	Trusted Computer System Evaluation Criteria
TDI	Trusted Database Interpretation

TDS	Tabular Data Stream Protocol
TEF	TTAP Evaluation Facility
TFM	Trusted Facility Manual
TRB	Technical Review Board
TS	Transactions Serialized
TSQL	Transact-SQL
TTAP	Trust Technology Assessment Program
URL	Universal Record Locator
VGA	Video Graphics Array
VM	Virtual Machine

16 APPENDIX G - Bibliography and References

16.1 U. S. Government

1. Department of Defense, *Trusted Computer System Evaluation Criteria*, DOD 5200.28 STD, December 1985.

2. National Computer Security Center, *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria*, NCSC-TG-021, Version 1, April 1991.

3. National Computer Security Center, PAT Guidance Working Group, *Form and Content of Vendor Design Documentation*, May 1994.

4. National Computer Security Center, PAT Guidance Working Group, *Form and Content of Vendor Test Documentation*, National Computer Security Center, May 1994.

5. National Computer Security Center, *Interpretations for the TCSEC*

6. National Computer Security Center, *Derived Verification Requirements for TCSEC Class C2*: *Controlled Access Protection (Operating Systems and Database Management Systems*, Version 2.0, April 1997, http://www.radium.ncsc.mil/tpep/ttap/DBMS.DVR.html

7. National Computer Security Center, *Evaluation Process for C2 Products: Pilot Phase Version 1.0, January 1997*, http://www.radium.ncsc.mil/tpep/ttap/Process.html

8. National Computer Security Center, *Final Evaluation Report: Sybase, Inc.* SQL Server Version 11.0.6 and Secure SQL Server Version 11.0.6, CSC-EPL-96/002 and CSC-EPL-96/003, 3 March 1997.

16.2 Sybase

9. Sybase, Inc., Sybase Adaptive Server Anywhere 7.0.0 C2 Supplement , July 2000.

10. Sybase, Inc., Sybase Adaptive Server Anywhere 7.0.0 C2 Document Update, July 2000.

11. Sybase, Inc., *Adaptive Server™ Anywhere Reference Manual*, Version 7.0.0, Document ID: MC0058, March 2000.

12. Sybase, Inc., *Adaptive Server™ Anywhere User's Guide*, Version 7.0.0, Document ID: MC0057, March 2000.

13. Sybase, Inc., *Adaptive Server™ Anywhere Programming Interfaces Guide*, Version 7.0.0, Document ID: MC0059, March 2000.

14. Sybase, Inc., *Adaptive Server™ Anywhere Getting Started*, Version 7.0.0, Document ID: MC0056, March 2000.

15. Sybase, Inc., *Introducing SQL Anywhere Studio*, Version 7.0.0, Document ID: MC0055, March 2000.

16. Sybase, Inc., *Adaptive Server™ Anywhere Reference for JDK 1.1*, Version 09/23/99.

17. Sybase, Inc., *SQL Anywhere Fundamentals*, Training Materials, PB5-ED08-1

18. Sybase, Inc., *Mobile Computing with SQL Anywhere*, SA5-ED02-2, March 1997.

19. Sybase, Inc., Adaptive Server Anywhere C2 Test Plan.

20. Sybase, Inc., *Adaptive Server Anywhere C2 Test Matrices.*

21. Sybase, Inc., *Adaptive Server Anywhere C2 Test Procedures*

22. Sybase, Inc., *Adaptive Server Anywhere C2 Test Procedures [Integrated Login]*

23. Sybase, Inc., *DBTEST Documentation.*

16.3 Microsoft Windows NT

16.3.1 Documents:

24. Science Applications International Corporation; *FINAL EVALUATION REPORT: Microsoft Corporation, Windows NT Workstation and Server, Version 4.0, Service Pack 6a, with C2 Update*; 15 December 1999.

25. Microsoft Corporation, *C2 Administrator's and User's Security Guide, Microsoft Windows NT*, Version 4.0.

16.3.2 Books:

26. *Microsoft Windows NT Workstation Resource Kit*, Microsoft Press, 1996. (Referenced by the TFM.)

27. Stinson, Craig and Carl Siechert, *Running Microsoft Windows NT Workstation Version 4.0*, Microsoft Press, 1996.

28. *Administering Microsoft NT 4.0 Class Pack*, Microsoft Official Curriculum, 1998.

29. Solomon, David A., *Inside Windows NT*, Second Edition, Microsoft Press, 1998.

30. Sutton, Stephen A., *Windows NT Security Guide*, Trusted System Services, Inc., 1997.

16.3.3 Microsoft Web Sites:

Microsoft Website Link	Description
http://msdn.microsoft.com	Welcome to MSDN Online
http://support.microsoft.com/servicedesks/fileversion/default.asp?vartarget=msdn	Microsoft DLL Help database
http://msdn.microsoft.com/library/default.htm	Welcome to the MSDN Library
http://www.microsoft.com/ntserver/nts/downloads/recommended/SP6/allSP6.asp	Windows NT 4.0 Service Pack 6a
http://support.microsoft.com/support/kb/articles/Q246/0/09.ASP	Windows NT 4.0 Service Pack 6a Available
http://support.microsoft.com/support/kb/articles/Q241/2/11.ASP	List of Bugs Fixed in Windows NT 4.0 Service Pack 6/6a (Part 1)
http://support.microsoft.com/support/kb/articles/Q244/6/90.ASP	List of Bugs Fixed in Windows NT 4.0 Service Pack 6/6a (Part 2)
http://support.microsoft.com/support/kb/articles/q244/5/99.asp	Fixes Required in TCSEC C2 Security Evaluation Configuration for Windows NT 4.0 Service Pack 6a
http://support.microsoft.com/support/kb/articles/q240/0/49.asp	How to Obtain Diagnostic i386 Processor Integrity Tests Required by the C2 TFM for Windows NT 4.0
http://www.microsoft.com/security/issues/deployingc2.asp	Deploying Windows NT 4.0 in a C2 Evaluated Configuration
http://support.microsoft.com/support/kb/articles/Q190/2/15.ASP	INFO: MDAC-Related White Papers (ODBC/OLE DB/ADO/RDS)
http://www.microsoft.com/TechNet/win2000/win2ksrv/technote/sspi2k.asp	Security Support Provider Interface (SSPI)